ISSN 2226-0358

Contributions to the Proceedings of

ESRF

ICALEPCS 2011

13th International Conference on Accelerator and Large Experimental Physics Control Systems
10-14th October 2011, WTC Grenoble, France.

Foreword

The 13th International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS 2011) conference was held from the 10th October to the 14th October at the World Trade Center (WTC) in Grenoble, France. It was hosted by the European Synchrotron Radiation Facility (ESRF). The conference was endorsed by the European Physics Society/Experimental Physics Control Systems (EPS/EPCS), the Institute of Electrical and Electronics Engineers (IEEE), Nuclear and Plasma Sciences Society (NPSS), the French Society of Physics (SFP Interdivision Physique des Accélérateurs et Technologies Associées), Physical Society of Japan (JPS), Particle Accelerator Society of Japan (PASJ), and the Association of Asia Pacific Physical Societies (AAPPS). The conference's Local Organising Committee was advised by two committees made up of members from the three regions – the International Scientific Advisory Committee (ISAC) and the Programme Committee (PC). The ISAC was comprised of 47 members and chaired by Roland Mueller of HZB. The PC was made up of 39 members and chaired by Jean-Michel Chaize of the ESRF.

ICALEPCS 2011 had a record attendance. In total 625 registered. Of these 415 were delegates, 54 were 1 or 2 day only attendees, 38 were students or retirees, 18 were sponsors or VIPs, 21 were accompanying persons, 31 were editors or assistants. CERN was the most represented institute followed by the ESRF. The breakdown of attendees by region is as follows: 53 for Asia and Oceania, 451 For Europe, Middle East and Africa, 99 for North and South America.

A total of 573 abstracts were submitted of which 171 were withdrawn. Out of the remaining 402 abstracts 112 were awarded oral status (this includes the keynote and invited talks) and 290 were posters. Mini-orals were introduced for the first time at ICALEPCS. During the mini-oral the speaker was given 3 minutes to present their poster. 31 posters were presented as mini-orals.

ICALEPCS 2011 put strong emphasis on having a high number of quality keynote speakers. For the first time 7 specialists from outside the community presented keynote talks. The feedback on the keynotes from the attendees was very positive proving they clearly contributed to the success of the conference in terms of the quality of the conference and attracting attendees.

The conference themes were chosen by the ISAC. The scientific programme was drawn up by the PC. Both committees did an excellent job choosing topics and talks and encouraging experts to present their work. A number of institutes were represented at ICALEPCS for the first time. Parallel sessions were reintroduced. Out of a total of 20 sessions, 7 were parallel.

There were 6 pre-conference workshops held on the Sunday before the conference. The workshop themes were: Open Hardware Initiative, Cyber-Security, TANGO, jddd, and CDM (Common data model). The attendance of the pre-conference workshops was again very high. Over 200 people attended the workshops.

Tutorials were reintroduced at ICALEPCS 2011. Two tutorials were given: Control Theory and Application to Accelerators and Fusion Reactors by Stefan Simrock and **Implementing DSLs with Xtext and MPS** by Markus Völter. The tutorials were very well attended and much appreciated by the attendeees. A round table discussion was held with some keynote speakers on *When and How to mix Languages. If not why not?*

For the second time the ICALEPCS Lifetime Achievement Award (LAA) was given. The ICALEPCS 2011 award was presented to: Emmanuel Taurel (ESRF), Nicolas Leclerq (SOLEIL), and Pascal Verdier (ESRF) on behalf of and for their contribution to the TANGO collaboration over the last 10 years.

For the first time a selection of the ICALEPCS papers will be published in a peer reviewed journal: Physical Review Special Topics Accelerators and Beams (PRST-AB). A special edition of the journal will be published in mid 2012. All papers accepted and presented at ICALEPCS 2011 are eligible to submit for publication in the peer reviewed journal.

The local organisation of the conference was under the responsibility of the Local Organising Committee (LOC). The LOC was comprised of 9 people and was chaired by Anne-Françoise Maydew. The LOC put emphasis on complementing the high quality scientific programme with an excellent social programme. The conference started with a welcome cocktail in the modern art museum hosted by the city of Grenoble. On the second evening conference attendees were treated to typical French culinary specialties (cheese and wine) and dancing (including the famous French can-can!). An outing was organised during the conference to the Château de Vizille (an important event happened there during the French Revolution). The gala evening was organised in an unusual venue – the Grenoble ice rink - where attendees were treated to a world class French magician show. The post conference tour was a trip to Chamonix and a ride with the cable car up to the Aiguille de Midi to marvel at the spectacular scenery of the Mont Blanc mountain range from up-close.

The ISAC accepted the bid for ICALEPCS 2015 to be held in Melbourne (Australia) hosted jointly by the Australian Synchrotron and the Australian Nuclear Science and Technology Organisation (ANSTO).

As chairman I would like to thank everyone who contributed to make ICALEPCS 2011 a successful and memorable event. The conference was the result of many people's work and support both financial and moral. It all started in 2007 in Knoxville when the ISAC of ICALEPCS 2007 accepted the ESRF bid to host ICALEPCS 2011 in Grenoble. It continued with ICALEPCS 2009 who decided to sponsor ICALEPCS 2011 with the earnings that remained over from ICALEPCS 2009. I thank all the sponsors and exhibitors who helped finance the conference. The ISAC chair and committee did a great job advising the LOC during the 2 years leading up to the conference. The PC

did an excellent job shaping the scientific programme. The editors did a huge and excellent job editing the many abstracts and papers to ensure that the papers were of high quality. The delegates made the conference a success by participating with high quality talks and questions. The city of Grenoble, the WTC and the event organisers (Insight Outside) made a special effort to help make ICALEPCS a success and make the delegates feel welcome. A big thanks to the ESRF, the host institute, and all the colleagues who helped and provided support for ICALEPCS 2011.

Finally I would like to especially thank (1) the proceedings editor, Marie Robichon, for setting up and maintaining the SPMS and ensure that the Proceedings were ready on time, and (2) the LOC chair, Anne-Françoise Maydew, both of whom went far above their normal call of duty, and (3) all the LOC members who worked very hard to ensure ICALEPCS 2011 was a success!

I look forward to seeing everyone in San Francisco in 2013 and in Melbourne in 2015!

The Conference Chair

Andy Götz

Contents

International Scientific Advisory Committee

Roland MÜLLER (ISAC Chair) - HZB, Berlin, Germany
Andy GÖTZ (Conference Chair) - ESRF, Grenoble, France
Jean-Michel CHAIZE (Programme Chair) – ESRF, Grenoble, France
Steve AZEVEDO - NIF, Livermore, USA
Ralph BAER – GSI, Darmstadt, Germany
Vito BAGGIOLINI - CERN, Geneva, Switzerland
Pascale BETINELLI - SOLEIL, Gif sur Yvette, France
Eric BJÖRKLUND - LANL, Los Alamos, USA
Alain BUTEAU - SOLEIL, Gif sur Yvette, France
Gianluca CHIOZZI - ESO, Garching, Germany
Peter CLOUT - VISTA, Los Alamos, USA
Bob DALESIO - NSLS II, Brookhaven, USA
Philip DUVAL – DESY, Hamburg, Germany
Laurent FARVACQUE - ESRF, Grenoble, France
Pravin FATNANI - RRCAT, Indore, MP, India
Kazuro FURUKAWA – KEK, Tsukuba, Japan
Philippe GAYET – CERN, Geneva, Switzerland
Nick HAUSER – ANSTO, Sydney, Australia
Mark HERON - DIAMOND, Didcott, UK
Larry HOFF - RHIC, New York, USA
In Soo KOO - POSTECH, Kyungbuk, Korea
Jörg KLORA - ALBA, Barcelona, Spain
Jean-March KOCH – ESRF, Grenoble, France
Ivan KOHLER - iThemba Labs, Faure, South Africa
Sharon LACKEY - Fermilab, Chicago, USA
Krister LARSSON - MAXLab, Lund, Sweden
Ge LEI - IHEP, Beijing, China
Patrick LE DÛ - IN2P3, Lyon, France
Marco LONZA – ELETTRA, Trieste, Italy
John MACLEAN – APS, Chicago, USA
Christopher MARSHALL – NIF, Livermore, USA
Mike MOUAT - TRIUMF, Vancouver, Canada
Niko NEUFELD – CERN, Geneva, Switzerland
Nhi Dien NGUYEN - NRI, Dalat, Vietnam
Stephane PEREZ - CEA/LMJ, Barp, France
James Rezende PITON – LNLS, Campinas, Brazil
Kay REHLICH - DESY, Hamburg, Germany
Claude SAUNDERS - APS, Chicago, USA
Javier SERRANO – CERN. Geneva, Switzerland
Grigori SHIRKOV – JINR, Dubna, Russia
Hamid SHOAEE – SLAC, San Jose, USA
Ryotaro TANAKA - SPRING8, Hyogo, Japan
Anders WALLANDER – ITER, St Paul lez Durance, France
Karen WHITE – SNS, Oak Ridge, USA
Noboru YAMAMOTO - KEK, Tsukuba, Japan

Programme Committee

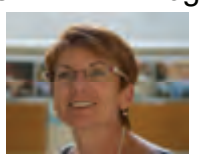
Jean-Michel CHAIZE (Programme Chair) – ESRF, Grenoble, France
Steve AZEVEDO - NIF, Livermore, USA
Ralph BAER – GSI, Darmstadt, Germany
Vito BAGGIOLINI - CERN, Geneva, Switzerland
Pascale BETINELLI - SOLEIL, Gif sur Yvette, France
Eric BJÖRKLUND - LANL, Los Alamos, USA
Alain BUTEAU - SOLEIL, Gif sur Yvette, France
Gianluca CHIOZZI - ESO, Garching, Germany
Laurent FARVACQUE - ESRF, Grenoble, France
Kazuro FURUKAWA – KEK, Tsukuba, Japan
Philippe GAYET – CERN, Geneva, Switzerland
Nick HAUSER – ANSTO, Sydney, Australia
Mark HERON - DIAMOND, Didcott, UK
Larry HOFF - RHIC, New York, USA
In Soo KOO - POSTECH, Kyungbuk, Korea
Jörg KLORA - ALBA, Barcelona, Spain
Jean-March KOCH – ESRF, Grenoble, France
Sharon LACKEY - Fermilab, Chicago, USA
Krister LARSSON - MAXLab, Lund, Sweden
Gongfa LIU - NSRL, Hefei, China
Patrick LE DÛ - IN2P3, Lyon, France
Marco LONZA – ELETTRA, Trieste, Italy
John MACLEAN – APS, Chicago, USA
Mike MOUAT - TRIUMF, Vancouver, Canada
Mikyung PARK – NFRI, Daejon, South Korea
Niko NEUFELD – CERN, Geneva, Switzerland
Stephane PEREZ - CEA/LMJ, Barp, France
Harshad PUJURA – ITER, St Paul lez Durance, France
Kay REHLICH - DESY, Hamburg, Germany
Claude SAUNDERS - APS, Chicago, USA
Javier SERRANO – CERN. Geneva, Switzerland
Liren SHEN – SSRF, Shangai, China
Hamid SHOAEE – SLAC, San Jose, USA
Wojtek SLIWINSKI - CERN, Geneva, Switzerland
Ryotaro TANAKA – SPRING8, Hyogo, Japan
Benjamin TODD – CERN, Geneva, Switzerland
Anders WALLANDER – ITER, St Paul lez Durance, France
Karen WHITE – SNS, Oak Ridge, USA
Markus ZERLAUTH – CERN, Geneva, Switzerland

Local Organising Committee

Anne-Françoise MAYDEW - Local Organising Committee Chair

Andy GÖTZ - Conference Chair

Jean-Michel CHAIZE - Programme Chair



Marie ROBICHON - Conference Proceeding's Editor

Fabienne MENGONI - Conference Office

Rudolf DIMPER – Sponsors' Coordinator

Staffan OHLSSON - Technical Support

Jens MEYER - Posters

Rainer WILCKE – Posters

Proceedings Editors

Marie ROBICHON, ESRF

Cindy CASSADY, LLNL

Carl FINLAY, CLS

Lynda GRAHAM, ESRF

Michaela MARX, DESY

Raphael MÜLLER, GSI

Maria POWER, ANL

Volker RW SCHAA, GSI

Thomas THUILLIER, LPSC

Emmanuelle VERNAY, LPSC


We made it!
Pre-conference only = 27
Delegates = 392
1/2 days = 63
Students = 61
VIPs = 11
Retirees = 3
Exhibitors = 43
600!
Accompanying persons = 20
Organisation = 19
Editors = 9
icalepcs 2011
icalepcs 2011


12 min


WORLD TRADE CENT
CHAMBRE DE COMMERCE & D'INDUSTRIE DE GRE
WORLD TRADE CENTER

More photos courtesy of Lillian Cardonne can be found on the conference web site

and courtesy of Stéphane Vallet can be found at this web address:

http://accelconf.web.cern.ch/AccelConf/icalepcs2011/photo-album/index.htm

Acknowledgments

ICALEPCS 2011 could not have been possible without the help and support of many people. We would like to acknowledge the following people and organisations:

ESRF for hosting the conference and providing the resources to organise it.

The members of the ISAC for choosing the session topics and providing valuable advice.

The members of the PC for setting up an interesting programme.

The proceedings editors – Cindy Cassady (LLNL), Carl Finlay (CLS), Lynda Graham (ESRF), Michaela Marx (DESY), Raphael Müller (GSI), Maria Power (ANL), Marie Robichon (ESRF), Volker RW Schaa (GSI), Thomas Thuillier (LPSC), Emmanuelle Vernay (LPSC).

The author receptionists: Debbie Davison (ESRF), Kirstin Colvin (ESRF)

Volker RW Schaa for preparing the abstracts for the conference guide and for assisting in the preparation of the proceedings.

The helpers who assisted the speakers during the conference.

The numerous ESRF staff who helped with the conference directly and indirectly.

The following learned societies for endorsing the conference – European Physical Society / Experimental Physics Control Systems (EPS/EPCS), Institute of Electrical & Electronics Engineers (IEEE), Nuclear & Plasma Science Society (NPSS), French Society of Physics (SFP Interdivision Physique des Accélérateurs et Technologies Associées), Physical Society of Japan (JPS), Association of Asia Pacific Physical Societies (AAPPS).

The following sponsors for providing financial support to the conference:
Platinum sponsors: SPRING8, Ville de Grenoble;
Gold sponsors: Grenoble-Alpes Métropole, Région Rhône-Alpes, National Instruments, NetApp, SOLEIL;
Image sponsors: Codaralp, Sodexo, Ricoh;
Delegates' bags: Cosylab, National Instruments;
Exhibitors: Aerotech Ltd, Beckhoff Automation GmbH, Caen, Delta Tau UK Limited, Diamond Detectors Ltd, DSoFt Solutions Ltd, D-Tacq Solutions, Friatec AG, FuG Elektronik GmbH, Heason Technology Ltd, Incaa Computers, Instrumentation Technologies, Magenta Systems for Agilent, Micro Controle Spectra Physics, Nexeya, Physical Instruments, Serviware, Struck Innovative Systeme GmbH.

The Grenoble Bureau des Congrès and Tourist Office for their support and help while preparing the conference and during the conference.

NEWS FROM ITER CONTROLS – A STATUS REPORT

A.Wallander, L. Abadie, F. Di Maio, B. Evrard, J-M. Fourneron, H. Gulati, C. Hansalia, J-Y. Journeaux, C. Kim, W-D. Klotz, K. Mahajan, P. Makijarvi, Y. Matsumoto, S. Pande, S. Simrock, D. Stepanov, N. Utzel, A. Vergara, A. Winter, I. Yonekawa,
ITER Organization, Route de Vinon sur Verdon, 13115 Saint Paul Lez Durance, France

Abstract

Construction of ITER has started at the Cadarache site in southern France. The first buildings are taking shape and more than 60 % of the in-kind procurement has been committed by the seven ITER member states (China, Europe, India, Japan, Korea, Russia and United States). The design and manufacturing of the main components of the machine is now underway all over the world. Each of these components comes with a local control system, which must be integrated in the central control system. The control group at ITER has developed two products to facilitate this; the plant control design handbook (PCDH) and the control, data access and communication (CODAC) core system. PCDH is a document which prescribes the technologies and methods to be used in developing local control systems and sets the rules applicable to the in-kind procurements. CODAC core system is a software package, distributed to all in-kind procurement developers, which implements the PCDH and facilitates the compliance of the local control system. In parallel, the ITER control group is proceeding with the design of the central control system to allow fully integrated and automated operation of ITER. In this paper we report on the progress of the design and technology choices and we discuss justifications of those choices. We also report on the results of some pilot projects aimed at validating the design and technologies.

INTRODUCTION

The first concrete was poured in the ITER tokamak seismic pit in August 2011. Construction of the first two buildings is expected to be completed next year and the control building in 2015. In other words, the civil construction of ITER is in full swing. At the same time, more than 60 % of the value of in-kind procurements has been committed by the seven member states and design and construction of ITER components have started all over the world. These components of ITER, procured in-kind, come with their local control systems.

The main challenge for ITER control system is to integrate all these local control systems and enable integrated and automatic operation of the complete ITER facility. Since the ITER schedule is driven by specifications of in-kind procurements the priority of the ITER control group has been to establish the standards and technologies affecting the local control systems as well as facilitating future integration. Two evolving products have been developed to allow this: a set of documents, called the Plant Control Design Handbook (PCDH), defining the standards and a control system framework, called CODAC (Control, Data Access and Communication) Core System, implementing those standards and providing a development environment for local control systems. In parallel, the design of the central system is proceeding with the preliminary design review scheduled for late 2011.

Plant Control Design Handbook

The PCDH is a set of documents that defines mandatory rules, recommended guidelines, methodologies and catalogues of supported products. It is a living document with the latest release issued in February 2011. The PCDH specifies the design methodology for development of local control systems including deliverables. The product catalogue includes Siemens Simatic S7 PLC, IEI PICMG 1.3 PC, National Instrument multipurpose PXI6259 I/O board, Sarel cubicles and more.

The PCDH is contractually binding in all in-kind procurements, which include local control systems. It has been subjected to thorough review by representatives of all ITER member states. The ITER control group is actively promoting PCDH by organizing presentations and training in the member states as well as developing pilot cases for proof of concept. PCDH is available at [1].

CODAC Core System

CODAC Core System (CCS) is a control system framework that implements the standards defined in the PCDH and guarantees that the local control systems can be integrated into the central control system [2]. It runs on all computers within the ITER architecture, both locally and centrally. The most important feature is the use of EPICS [3], which guarantees communication using the channel access protocol. Major releases of CCS are made on a yearly basis. ITER contributions on top of EPICS are publicly available at [1].

Organizations contributing to the ITER project, in particular developers of local control systems can become registered users. A registered user is provided with support, such as software distribution, help desk and access to the development environment.

Before starting to discuss the chosen technologies and the reason for those choices, we briefly describe the overall architecture of the ITER control system. A more detailed description of the architecture is available in [4].

ARCHITECTURE

The architecture of the ITER control system is illustrated in Fig. 1. The main principles are maintained from the conceptual design [5], segregation in three vertical tiers; conventional control, interlock and safety and two horizontal layers; central and local. The local layer is procured in-kind, while the central layer is developed by the host, ITER Organization in France.

The current estimate is that there will be 220 local systems, grouped together and organized into 18 ITER subsystems. These local systems consist of a set of controllers, interfacing the actuators and sensors, and connected together via network switches to the plant operation network (PON). A similar architecture is applied for interlocks (CIN) and safety (CSN). Coordination and orchestration at the central level are first done at the subsystem level and then at the central supervision level. The human machine interface is provided by dedicated CODAC terminals located in the control room or close to the equipment for commissioning and troubleshooting purposes. For the safety system there is a dedicated safety desk in the main and backup control rooms.

In addition, a number of dedicated networks are used; TCN for time distribution and synchronization, SDN for real-time feedback and asynchronous events, ARCN for scientific data archiving and AVN for video transmission.

Table 1: ITER Controls Main Parameters

Parameter	Value
Total number of computers	1.000
Total number of signals (wires)	100.000
Total number of process variables	1.000.000
Total number of active operator screens	100
Update rate per screen (200 PVs)	5 Hz
Maximum sustained data flow on PON	50 MB/s
Total engineering archive rate	5 MB/s
Total scientific archive rate (initial)	1 GB/s
Total scientific archive rate (final)	20 GB/s
Total scientific archive capacity	Few PB/year
Accuracy of time synchronization	50 ns RMS
Number of nodes on SDN	100
Maximum latency asynchronous events	1 ms
Maximum latency sensor to actuator (SDN)	500 µs
Maximum jitter sensor to actuator (SDN)	50 µs
Maximum sustained data flow on SDN	25 MB/s

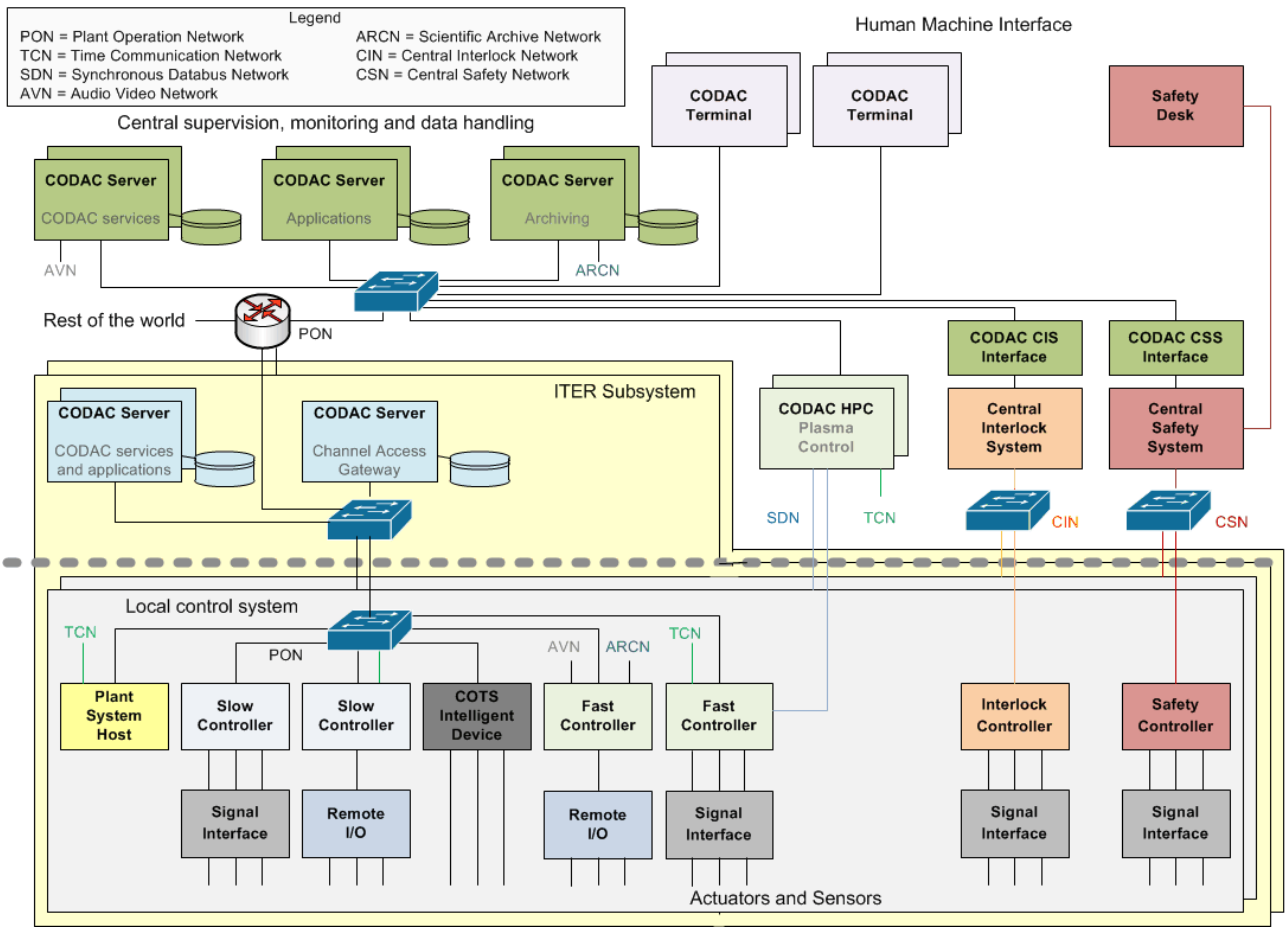


Figure 1: Physical architecture of ITER control system.

SELECTED TECHNOLOGIES

EPICS

A key technology decision was taken in 2009 with the adoption of the EPICS toolkit. This decision gave a head start to the development of CODAC Core System because all key requirements, such as robust communication and "live database", are solved by EPICS. In the preceding evaluation, open source solutions took preference over commercial ones because of longevity requirements. Of the open source alternatives EPICS was the clear winner because of its proven track record of reliability and scalability and the fact that it has been successfully deployed in almost all ITER member states, including on tokamaks in Korea (KSTAR) and the US (NSTX). CODAC Core System v2 comes with EPICS core v 3.14.12 and a selection of additional EPICS packages.

Linux

A second important decision was to adopt Linux as the base operating system. It is an obvious choice considering the selection of EPICS and the general market of large experimental facilities in the world. An evaluation was carried out of the three major commercial providers of Linux: Ubuntu, Red Hat and SUSE. Red Hat was selected for being the market leader and providing the longest support of major releases. CODAC Core System v2 comes with Red Hat Enterprise Linux (RHEL) v5.5 x86_64. It is planned to upgrade to v6.1 in the next major release due in early 2012.

It is estimated that more than 80 % of systems present in the entire control system are non-real-time or soft real-time. The remaining 20 % require real-time features not provided by RHEL. After an evaluation and benchmark of alternatives we have selected MRG-R from Red Hat.

Control System Studio

Control System Studio (CSS), developed by DESY, ORNL and Brookhaven, is an Eclipse-based collection of tools running on top of EPICS to monitor and operate large scale control systems. An evaluation was carried out to address both the functionality and performance of the product and match the requirements, mainly on the presentation layer, of CODAC. Many of these tools rely on a relational database (RDB), typically Oracle or MySQL. However, due to the uncertainty of MySQL's future as open source ITER has selected PostgreSQL as RDB. The evaluation therefore also included compatibility tests with PostgreSQL. The result was to adopt BOY (Best ever Operating interface Yet), BEAST (Best Ever Alarm System Toolkit), BEAUTY (Best Ever Archive UTility, Yet) and SNL Editor (State Notation Language Editor) in CODAC Core System.

Configuration

To address concerns related to the development of local control systems in all member states, an in-house development in unifying configurations has been undertaken. The basic idea is to capture configuration data like definition of process variables, input/output configuration, alarm definitions etc. in a relational database. These data are then used to auto-generate configuration files such as EPICS db files, BEAST and BEAUTY configuration files, BOY engineering screens, driver configurations etc. Further details of this project, called self-description data and using Hibernate, Spring and Eclipse technologies, are given in [6].

Development Environment and Distribution

The distributed nature of the ITER control system development (Fig. 2) requires high quality, both on the product (CODAC Core System) and the software distribution. To achieve quality and be cost effective unifying processes and technologies are essential. Major investments have been made in this area. Subversion has been selected as the version control tool, Apache Maven as the build tool, RPM (Red Hat package management) for packaging and for managing dependencies, Bugzilla as the issue tracking system, Jenkins for continuous integration and automated test execution and Red Hat Network Satellite Server for software distribution [7].

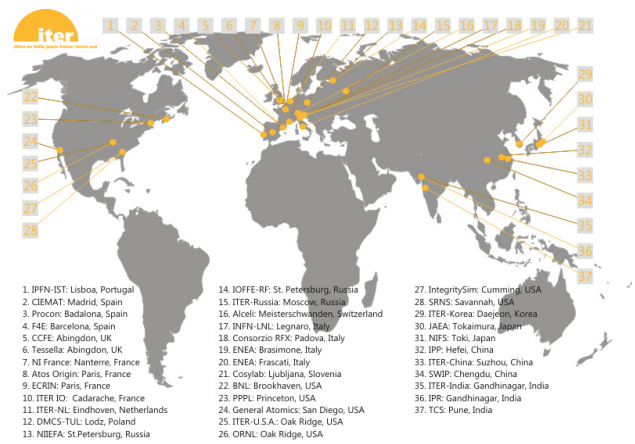


Figure 2: Registered organizations using CCS.

PTPv2 (IEEE 1588-2008)

The driving requirement for synchronizing ITER computers is the off-line correlation of different diagnostics data using absolute time stamps. The requirement has been set to 50 ns RMS. Requirements for pre-programmed triggers are much more relaxed (1 ms RMS). These requirements are matched closely by the precision time protocol (IEEE 1588-2008) now widely supported by many vendors. We have implemented the CODAC TCN network based on a PTPv2 using Meinberg and Symmetricom grand master clocks, IEEE 1588-2008 compatible industrial switches (Cisco IE3000 and Hirschman MAR1040) and NI PXI 6682 timing receivers as host nodes and confirmed the performance.

10 GbE Multicast UDP

The most important requirement for the real-time network SDN is to support control loops with a cycle times of 500 µs, which includes acquisition, computation,

communication and actuation. Assuming a two hop configuration and that communication can use a maximum of 10 % of the available time budget yields an upper limit for latency between two nodes of 25 µs with a maximum jitter of 10 %. Such requirements are traditionally solved by reflective memory technologies. However, with the emergence of 10 Gb Ethernet, cut-through switches and wide support of multicast, switched Ethernet provides an alternative. We commissioned a benchmark with a major switch supplier confirming that switch latency for a cut-through switch is below 2 µs independent of package size. Using 10GbE NIC cards with protocol stack accelerators (Solarflare, Chelsio, Mellanox) we performed additional tests to measure latency including the UDP stack and the API. The results confirm that performance is comparable to reflective memory technology (Fig. 3). Considering technology evolution of Ethernet we have therefore selected 10GbE UDP multicast as our solution for the real-time network.

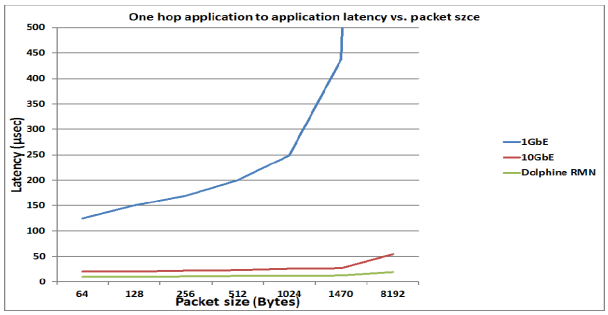


Figure 3. Measured latency 1GbE, 10GbE and RMN.

CURRENT R&D

The two biggest areas of technology selection not yet addressed are scientific archiving and plasma feedback control. A preliminary study of scientific data format has been carried out resulting in indications that HDF5 is the preferred option. This will be followed by the implementation of an archive prototype in 2012. Different options for high performance file systems are also being benchmarked, the main candidates being NFS4, HDFS and pNFS.

A major world-wide effort task with a consortium of experts in plasma control has been initiated. This task aims to detail the requirements and build engineering models of plasma control for ITER. In parallel an evaluation of existing real-time frameworks is underway. This framework, together with the real-time network, will make up the plasma control infrastructure. A primary candidate at this time is MARTe or some derivate of that.

A third area of active R&D concerns fast controllers, particular xTCA and FPGA technologies targeting diagnostics. A number of prototypes are being implemented with the aim of bringing such products into the PCDH catalogue of standard components.

PILOT PROJECTS

A number of pilot projects have been initiated to confirm the technology selections, prove the feasibility of implementing ITER controls using the selected approaches and identify any weak points to be addressed.

The ITER construction site and office buildings are currently powered by a 15kV substation. Applying the methodology and standard components defined in the PCDH, this substation has been successfully interfaced to ITER CODAC using two Siemens S7 PLCs, PSH, CODAC servers and CODAC Core System to monitor, archive and handle alarms for 400 process variables.

The neutral beam test bed facility in RFX Padova, Italy, will be used to test and commission the neutral beam injectors for ITER. The first system to be implemented will be an ion source and RFX has decided to apply CODAC Core System in this project. An initial evaluation has not shown any show-stoppers. The project is expected to be completed in 2015.

The Frascati Tokamak Upgrade (FSU) in Italy decided to upgrade the control for their flywheel generator using PCDH and CCS. This has been accomplished and an interface to the legacy control system has been implemented. Commissioning of the system during tokamak operation is planned in the autumn.

The control system of KSTAR (Korea Superconducting Tokamak Advanced Research) is based on EPICS and therefore a perfect candidate for testing ITER CODAC concepts. A collaboration project is underway to convert the hydrogen fuelling system to ITER standards and to implement closed-loop density control. The conclusion of the first phase is scheduled for late 2011 and of the second phase for late 2012.

CONCLUSIONS

The design and implementation of the ITER control system is progressing according to plan. Many technology decisions have been taken during the last year. The success or failure of the project will be determined by the acceptance of this work by the ITER member states responsible for providing in-kind local control systems.

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

[1] http://www.iter.org/team/chd/cid/codac
[2] F. Di Maio et al., “The CODAC software distribution for the ITER plant systems”, This conference
[3] http://epics.aps.anl.gov/epics/
[4] A. Wallander et al. “Baseline Architecture of ITER control system”, IEEE TNS vol. 58 1433-1438, Aug 2011
[5] J.B. Lister et al., “The status of the ITER CODAC conceptual design”, Fusion Engineering and Design 83 (2008)
[6] L. Abadie et al., “The self-description data configuration model”, IAEA TM8, San Francisco, June 2011
[7] A. Zagar et al., “ITER CODAC Core System Development Process”, IAEA TM8, San Francisco, June 2011

THE ATLAS DETECTOR CONTROL SYSTEM

S. Schlenker, S. Arfaoui, S. Franz, O. Gutzwiller, C.A. Tsarouchas, CERN, Geneva, Switzerland
B. Mindur, AGH University of Science and Technology, Krakow, Poland
J. Hartert, S. Zimmermann, Albert-Ludwig Universitaet Freiburg, Freiburg, Germany
A. Talyshev, BINP, Novosibirsk, Russia
D. Oliveira Damazio, A. Poblaguev, BNL, Upton, Long Island, New York, USA
H. Braun, D. Hirschbuehl, S. Kersten, K. Lantzsch, Bergische Universität, Wuppertal, Germany
T. Martin, P.D. Thompson, Birmingham University, Birmingham, UK
D. Caforio, C. Sbarra, Bologna University, Bologna, Italy
D. Hoffmann, CPPM, Aix-Marseille Université, CNRS/IN2P3, Marseille, France
S. Nemecek, Czech Republic Academy of Sciences, Prague, Czech Republic
A. Robichaud-Veronneau, DPNC, Geneva, Switzerland
B. Wynne, Edinburgh University, Edinburgh, UK
E. Banas, Z. Hajduk, J. Olszowska, E. Stanecka, IFJ-PAN, Krakow, Poland
M. Bindi, A. Polini, INFN-Bologna, Bologna, Italy
M. Deliyergiyev, I. Mandić, JSI, Ljubljana, Slovenia
E. Ertel, Johannes Gutenberg University Mainz, Mainz, Germany
F. Marques Vinagre, G. Ribeiro, H. F. Santos, LIP, Lisboa, Portugal
T. Barillari, J. Habring, J. Huber, MPI, München, Germany
G. Arabidze, MSU, East Lansing, Michigan, USA
H. Boterenbrood, R. Hart, NIKHEF, Amsterdam, Netherlands
G. Iakovidis, K. Karakostas, S. Leontsinis, E. Mountricha, National Technical University of Athens, Greece
V. Filimonov, V. Khomutnikov, S. Kovalenko, PNPI, Gatchina, Russia
V. Grassi, SBU, Stony Brook, New York, USA
J. Mitrevski, SCIPP, Santa Cruz, California, USA
P. Phillips, STFC/RAL, Chilton, Didcot, Oxon, UK
S. Chekulaev, TRIUMF, Vancouver, Canada
S. D'Auria, University of Glasgow, Glasgow, UK
K. Nagai, University of Tsukuba, Tsukuba, Ibaraki, Japan
G.F. Tartarelli, Université degli Studi di Milano & INFN, Milano, Italy
G. Aielli, F. Marchese, Università di Roma II Tor Vergata, Roma, Italy
P. Lafarguette, Université Blaise Pascal, Clermont-Ferrand, France
R. Brenner, Uppsala University, Uppsala, Sweden

Abstract

The ATLAS experiment is one of the multi-purpose experiments at the Large Hadron Collider (LHC), constructed to study elementary particle interactions in collisions of high-energy proton beams. Twelve different sub-detectors as well as the common experimental infrastructure are supervised by the Detector Control System (DCS). The DCS enables equipment supervision of all ATLAS sub-detectors by using a system of >130 server machines running the industrial SCADA product PVSS. This highly distributed system reads, processes and archives of the order of 10^6 operational parameters. Higher level control system layers allow for automatic control procedures, efficient error recognition and handling, and manage the communication with external systems such as the LHC. This contribution firstly describes the status of the ATLAS DCS and the experience gained during the LHC commissioning and the first physics data taking operation period. Secondly, the future evolution and maintenance constraints for the coming years and the LHC high luminosity upgrades are outlined.

INTRODUCTION

The ATLAS experiment [1] at the LHC aims to study the physics of high energy particle interactions in a previously unexplored energy domain. The detector elements are distributed over a cylindrical volume of 25 m diameter and 50 m length. More than 4000 people of 176 institutions in 40 countries contribute to the project.

The DCS has the task to permit coherent and safe operation of ATLAS and to serve as a homogeneous interface to all sub-detectors and the technical infrastructure of the experiment. The DCS must bring the detector into any desired operational state, continuously monitor and archive the operational parameters, signal any abnormal behavior. A more detailed description of the complete ATLAS DCS and specific sub-detector control system hardware and software can be found in [1, 2] and references therein.

OVERALL SYSTEM DESIGN

The DCS was designed and implemented within the frame of the Joint Controls Project (JCOP) [3], a collaboration of the CERN controls group and DCS teams of the LHC experiments. Standards for DCS hardware and software were established together with implementation guidelines both, commonly for JCOP and specifically for ATLAS.

The Front-End (FE) equipment consists of purpose-built electronics and their associated services such as power supplies or cooling circuits. For the implementation of the

DCS Back-End (BE), the industrial Supervisory Controls And Data Acquisition (SCADA) product *PVSS* serves as base software. On top of PVSS, the JCOP Framework facilitates the integration of standard hardware devices and the implementation of homogeneous controls applications. The BE is organized in three layers (see Fig. 1): process control of subsystems, a single control station for a sub-detector allowing stand-alone operation, and global stations with service applications and operator interfaces.

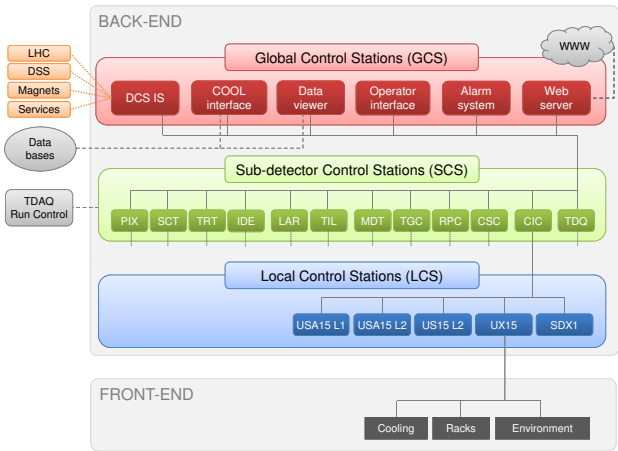


Figure 1: ATLAS DCS architecture.

SYSTEM STANDARD BUILDING BLOCKS

Front-End

The DCS FE equipment had to meet common requirements such as low cost, low power consumption, and high I/O channel density. For equipment interconnection, the CAN industrial field-bus and the CANopen protocol is used wherever possible and appropriate. Electronics in the detector cavern had to allow for remote firmware upgrades, be insensitive to magnetic fields, and be tolerant to radiation exposure expected during the experiment lifetime.

ELMB: A low-cost custom-built I/O concentrator, the Embedded Local Monitoring Board (ELMB) [4] was developed as common solution for interfacing custom designs to the DCS. The ELMB board ($50 \times 67\,\mathrm{mm}^2$) features a 8-bit 4 MHz micro-controller with 64 analog and 32 digital channels and a CAN bus interface. The board is tolerant to strong magnetic fields and radiation hard for integrated doses up to 50 Gy. Further, the ELMB can be embedded within custom designs and has a modular, remotely extendable firmware with a general purpose CANopen I/O application. More than 10000 ELMBs are in use within all LHC experiments, over 5000 alone within ATLAS.

Standardized Commercial Equipment: The industrial standard VME is used to house electronics. For all crates, monitoring is implemented for temperature and general status information as well as power and reset control. The detector components are powered by different types of industrial power supplies featuring control of voltages/currents, over-voltage/current protection, and thermal supervision.

Back-End

DCS Control Station PC: The hardware platform for the BE system are industrial, rack-mounted server machines. Two different standard machine types, one for applications requiring good I/O capability, a second for processing-intensive applications with I/O via Ethernet connectivity. Both models feature redundant, hot-swappable power supplies and disk shadowing.

PVSS: The SCADA package *PVSS* (re-branded to *SIMATIC WinCC OA*) is the main framework for the BE applications. Four main concepts of PVSS make it suitable for a large scale control system implementation:

- Generic types of control process templates may be used depending upon the type of the required application avoiding unnecessary overhead.
- Each PVSS application uses a local database for the storage of control parameters providing synchronized access for all connected processes. Data processing is performed with an event-based approach and data is made persistent by archiving selected DCS parameters to an external Oracle database.
- Different control systems can be connected via LAN to form a *Distributed System* allowing for highly scalable remote data access and event notification.
- A generic Application Programming Interface (API) allows to further extend the functionality of control applications.

Front-end interface software: For interfacing the front-end devices with PVSS, the industry standard OPC was chosen. Commercial equipment manufacturers as well as developers of custom devices provide the *OPC servers* for which PVSS provides an OPC client. For the ELMB CAN bus readout and control, a dedicated CANopen OPC server has been developed. Device types for which OPC could not be used due to maintenance or platform constraints (OPC is limited to MS Windows$^{\mathrm{TM}}$), custom readout applications were interfaced to PVSS using the CERN standard middle-ware *DIM* [5]. PLCs are interfaced to PVSS via Mod-Bus.

The Finite State Machine Toolkit: The JCOP FSM [6] provides a generic, platform-independent, and object-oriented implementation of a state machine toolkit for a highly distributed environment, interfaced to a PVSS control application. The attributes of an FSM object instance are made persistent within the associated PVSS application database. This allows for archiving of the FSM states and transitions, and integration of the FSM functionality into PVSS user interfaces.

INTEGRATION AND OPERATION

Control Hierarchy, Error Handling, Operation

The complete DCS BE is mapped onto a hierarchy of Finite State Machine (FSM) elements using the FSM toolkit. State changes are propagated upwards and commands downwards in the hierarchy allowing for the operation of the complete detector by means of a single

FSM object at the top level. A fixed state model (see Fig. 2) has been applied, reflecting detector conditions for which physics data taking is optimal (READY) or compromised (NOT_READY), or the detector has been turned off (SHUTDOWN). A special STANDBY state is reserved for detectors with intermittent stage for unstable beam conditions. The state UNKNOWN is used when the actual condition cannot be verified. TRANSITION signals a transient state, e.g. ongoing voltage ramps. The actual state of these logical objects is determined by the states of the associated lower level objects (children) via state rules. The lower level objects may follow a more sub-system-specific state model for which guidelines exist.

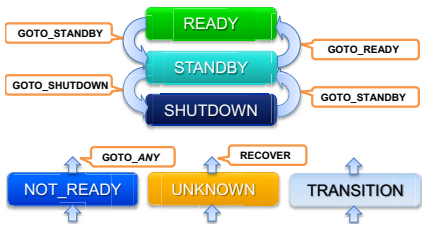


Figure 2: State model for high level objects.

For each critical parameter, alarms can be configured and are classified into one of the severity *Warning*, *Error*, or *Fatal*. To avoid the accumulation of a large number of alarms on the user interface, a masking functionality has been added to hide past occurrences e.g. after a follow-up has been initiated.

Each FSM object in the lowest hierarchy level has an attribute called *Status* which assumes the highest severity of alarms active for the respective device. The Status is then propagated up in the FSM hierarchy and thus allows for error recognition within the top layers of the detector tree and permits to identify problematic devices by following the propagation path downwards.

The DCS is operated from two primary, remotely accessible user interfaces – the FSM Screen for operation of the detector Finite State Machine hierarchy (see Fig. 3) and the Alarm Screen for alarm recognition and acknowledgment. Static status monitoring is provided by web pages on a dedicated web server allowing to quickly visualize all high level FSM user interface panels world-wide and without additional load of BE control stations.

Sub-Systems and LHC Interaction

The DCS of 9 main ATLAS sub-detectors, 3 forward detectors and common services have been integrated into a big distributed system with more than 10^7 individual parameters and subsequently condensed into approximately 10^5 state machine objects (see Table 1).

The data exchange between the ATLAS DCS and external control systems is handled via a dedicated, DIM-based data exchange protocol. All external control systems are homogeneously interfaced to the ATLAS DCS using dedicated DCS Information Servers. A generic data integrity monitoring has been implemented signaling any error condition related to the data quality and availability for the more than 20 different providers.

Table 1: Detector Sub-system Statistics. For each detector component, the # of server control stations and associated PVSS applications, the # of archived parameters, the total # of PVSS parameters, and the # of FSM objects are shown.

System	Component	#Servers (Appl.)	#Archived PVSS Parameters	#Total Parameters	#FSM Objects
Inner Detector	Pixel	11(12)	57k	1'086k	9.1k
	Silicon strips	11(11)	106k	1'265k	14.7k
	Transit. radiation	11(11)	69k	123k	13k
	Common services	7(8)	16k	494k	3.7k
Calorimeters	Liquid Argon	13(13)	27k	910k	8.3k
	Tile	5(5)	51k	719k	2.4k
Muon Spectrometer	Drift tubes	29(29)	214k	3'229k	19.2k
	Cathode strip	2(2)	1.3k	109k	0.6k
	Resistive plate	7(7)	139k	1'597k	2.5k
	Thin gap	7(7)	81k	1'225k	10k
	Common services	2(2)	0.7k	55k	0.04k
Forward detectors		4(4)	4.9k	194k	0.9k
Common Services	Counting rooms	7(7)	23k	568k	4.7k
	Trigger and DAQ	2(2)	11k	386k	1.3k
	External+safety	4(6)	8.0k	144k	0.4k
	Global services	9(13)	1.2k	222k	0.4k
Total		131(139)	809k	12.3M	91.2k

The interaction with the operational states of the LHC machine introduces a dynamic element into the operational model. During unstable beams phases (injection, ramp, etc.), the Silicon tracker and Muon detectors must remain at reduced voltage levels. This and additional beam-safety constraints require a hand-shake procedure with the LHC operators. Almost all beam-related actions – ramping the detector voltages depending on beam states with previous cross-checks on detector state and background rates – have been automated within DCS. This leaves DCS the full control over the LHC fill cycle, just requiring operator confirmation for beam injection and beam adjustments after stable beam periods. Finally, DCS is used for continuous monitoring of beam background rates and luminosity information during LHC fills. Figure 4 shows the evolution of typical LHC related parameters at the start of a fill such as beam energy and intensity, luminosity, and the change of voltage levels for two selected channels.

MAINTENANCE AND UPGRADES

Long-Term Maintenance and Organization

The DCS undergoes continuous consolidation, mostly driven by operational requirements, e.g. increasing automation for recurring problems such as power supply trip recovery or readout re-initialization after failures. As for the initial developments, common approaches are used as much as possible in order to limit the amount of potential implementation flaws and ease further maintenance. Some weak points requiring major effort during the future maintenance and consolidation include:

DCS control station upgrade: The usual life cycle of server machines of a maximum of 5 years requires hardware and operating system upgrades. A particular weakness is the choice of hardware I/O modules (e.g. CAN in-

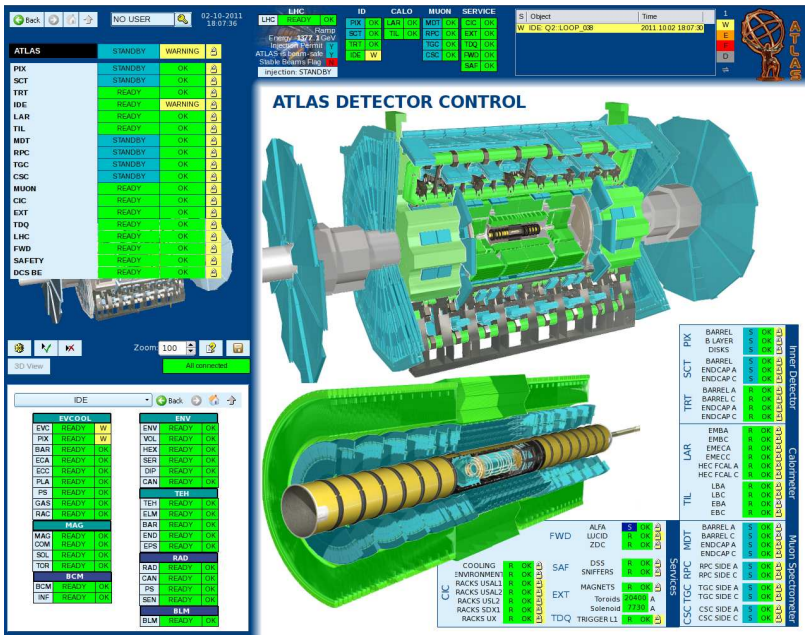


Figure 3: Operator interface (FSM Screen) showing the detector in *STANDBY* configuration during LHC ramp-up. The top hierarchy level object is shown together with its children objects (top left) and the associated main panel (bottom right). The UI allows the navigation to any FSM object and associated panel within the whole ATLAS DCS hierarchy. On the top right, a list of objects with non-OK Status allow shortcut navigation to problems.

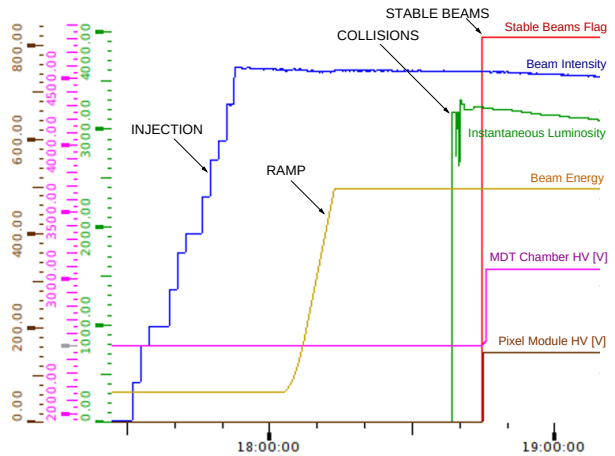


Figure 4: Initial phase of a LHC fill displayed using DCS parameters. The DCS response to the stable-beams signal received from the LHC is illustrated by two selected high voltage channels which are ramped to nominal values.

terface cards) using the PCI bus which have a short life cycle. In the future, USB and/or Ethernet interface adapters will be used which at the same time allow to increase the amount of bus connections per server machine. This enables the use of fewer cost-effective high-performance machines with a high number of cores running multiple DCS applications, or multiple virtual hosts.

Front-end interface software: The use of OPC allowed some degree of standardization of the FE interface software. However, it remains restricted to the Windows platform and lacks security mechanisms. Development activity has started to use OPC Unified Architecture (UA) – a platform independent successor of OPC with greatly increased flexibility, built-in security mechanisms, and the possibility to embed servers into electronics devices.

Future Upgrades

In the upcoming years, the LHC will undergo luminosity upgrades in several stages. Within the next 20 years, the maximum instantaneous luminosity will increase by a factor of 10 and the accumulated dose by a factor of 100 compared to the initial LHC design. This introduces more stringent requirements for DCS on-detector components such as the ELMB. A successor board has been proposed – the ELMB++ [7] – with improved radiation hardness and board flexibility, and removing previous weaknesses.

The ATLAS detector will undergo several upgrade stages for which new DCS components have to be designed and implemented. The first of these new projects is the DCS for an additional innermost barrel layer of the Pixel detector [8] which will be installed during the upcoming long shutdown in 2013/14.

REFERENCES

[1] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider", JINST 3 (2008) S08003
[2] A. Barriuso Poy et al., "The detector control system of the atlas experiment", JINST 3 (2008) P05006
[3] O. Holme et al., "The JCOP framework", ICALEPCS 2005, Conf.Proc.C051010:WE2.1-6O (2005)
[4] B. Hallgren wet al.,"The embedded local monitor board (ELMB) in the LHC front-end I/O control system", Stockholm 2001, Electronics for LHC experiments 325-330
[5] C. Gaspar et al., "DIM, a portable, light weight package for information publishing, data transfer and interprocess communication", Comput. Phys. Commun. 140 (2001) 102-109
[6] C. Gaspar and B. Franek, "Tools for the automation of large distributed control systems", IEEE Trans. Nucl. Sci. 53 (2006) 974-979
[7] S. Franz et al., "ELMB++, A Proposal for the Successor of the Embedded Local Monitoring Board", ATLAS CMS Electronics Workshop for LHC Upgrades (2011) poster
[8] S. Kersten et al., "Detector Control System of the ATLAS Insertable B-Layer", MOPMS021, ICALEPCS'11, Grenoble, France.

THE MEDAUSTRON ACCELERATOR CONTROL SYSTEM

J. Gutleber, CERN, Geneva, Switzerland
A. Brett, R. Moser, M. Marchhart, C. Torcato de Matos, EBG MedAustron, Austria
J. Dedic, Cosylab, Slowenia

Abstract

This paper presents the architecture and design of the MedAustron accelerator control system. This ion therapy and research facility is currently under construction in Wr. Neustadt, Austria. The accelerator and its control system are designed at CERN. This class of machine is characterized by rich sets of configuration data, real-time reconfiguration needs and high stability requirements. The machine is operated according to a pulse-to-pulse modulation scheme. Each beam cycle is described in terms of ion type, energy, beam dimensions, intensity and spill length. The control system is based on a multi-tier architecture with the aim to achieve a clear separation between front-end devices and their controllers. In-house developments cover a main timing system, a light-weight layer to standardize operation and communication of front-end controllers, fast and slow control of power converters and a procedure programming framework for automating high-level control and data analysis tasks.

INTRODUCTION

The MedAustron particle accelerator [1] is intended to deliver beams of various light ions for research and ion particle therapy. It features multiple ion sources, a Linac, a synchrotron and five beam lines including a proton-gantry (see Fig. 1). Upon request, virtual accelerators that represent beam paths from an ion source to an irradiation room supply beam cycles with pre-configured characteristics. A client system requests beam cycles with certain characteristics and the accelerator control system timely re-configures the front-end components. A request for a next beam cycle is issued while a cycle is generated (pipelined operation), requiring that the control system operates in soft real-time to keep dead-times between beam cycles low. Accelerator subsystems provide pulse-to-pulse modulation operation on a best effort basis, requiring a requesting client system to verify the characteristics of the delivered beam cycles.

Key features of the MedAustron control system are:

- Possibility to partition the accelerator into multiple, independent virtual accelerators and working sets to allow tuning of machine components and servicing concurrently with beam commissioning and operation.
- Optimized operation for pipelined pulse-to-pulse modulation supporting several hundreds of thousands of beam cycle configurations, being able to switch configuration in soft real-time without dead-time for cycle durations down to one second.
- High density remote power converter controller with sub-microsecond timing precision and value precision up to 24 bits operating at up to 50 KHz.
- Highly accurate and flexible real-time event distribution network as main timing system with 100 nanosecond GPS timestamping capabilities.
- Possibility to operate multiple machine partitions concurrently, supporting several modes and operation security levels.
- Role-based authentication and authorization.
- An RDBMS based system to store and organize configuration data and to create and trace versions of configuration data for operation.

This article gives an overview of the accelerator control system architecture and design decisions that govern its implementation.

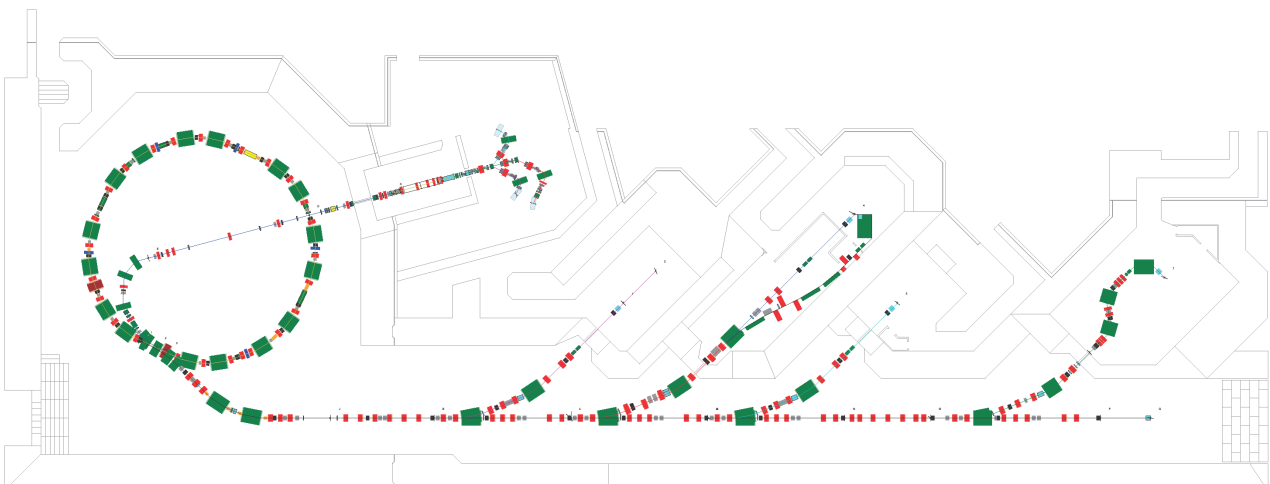


Figure 1: Layout of the MedAustron particle accelerator. Gantry beamline is indicated in bottom right part.

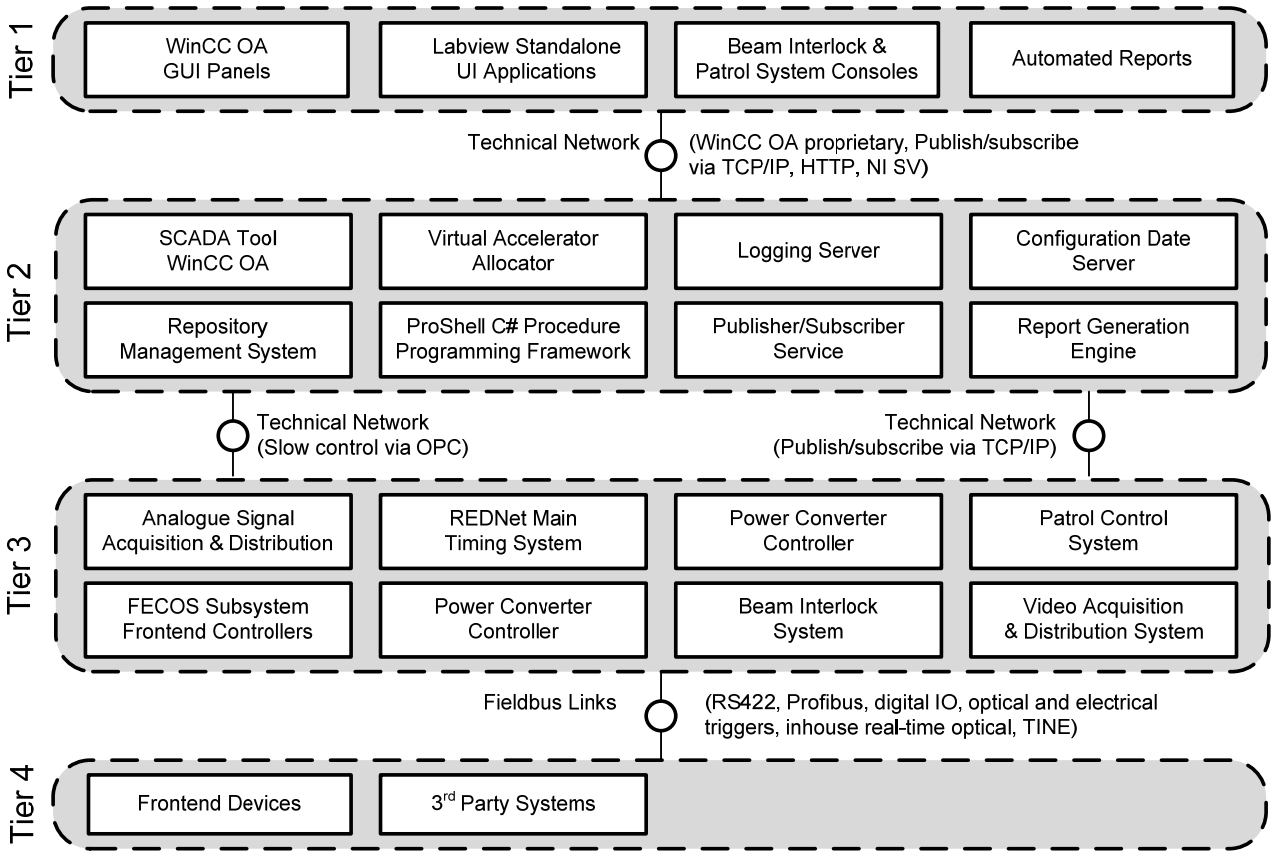


Figure 2: 4-tier accelerator control system architecture. Tier 4 comprises low-level front-end devices or third party control systems that must be integrated via a proxy front-end controller at tier 3. A single physical technical network permits communication among tiers in a secured network environment. Applications at tier 1 and 2 are virtualized.

ARCHITECTURE

Overview

The architecture (see Fig. 2) extends the industry best practice, 3-tier model [2,4] in accordance with [3]: (1) *presentation* tier, (2) *processing* tier, (3) *equipment* tier and (4) *frontend* tier. Components in separate tiers that are distributed over a number of processing devices communicate with each other through a dedicated Ethernet network. Communication between equipment tier and frontend tier may also be achieved through dedicated field-bus and custom links, depending on the imposed constraints. Subsequent sections describe each tier, starting with the frontend tier that is closest to the equipment under control.

Frontend Tier (T-4)

Front-end devices that control the actual accelerator hardware are categorized according to the project's work package organization:

The ion source supplier provides low-level API libraries for a COTS real-time computing platform. They serve implementing a front-end controller (FEC) at tier three that autonomously operates a single ion source. That system is replicated for each of the foreseen ion sources.

Conventional magnet temperature sensors open a circuit to indicate an over-temperature condition. A single PLC using the Siemens Safety Matrix tool [5] drives the circuits of all 262 magnets. This system interfaces also directly to all power converters to deliver machine protection interlock functions. Special magnets feature a separate, in-house developed, Siemens PLC based positioning and cooling monitoring system that interfaces directly to tier 2 via the S7 protocol over TCP/IP.

Power converters come in 2 different flavours: The first family is digitally controlled via RS422 lines and a purpose defined, UART based protocol for current control, slow controls and monitoring. All other power converters are treated as voltage sources that are controlled by in-house designed, analogue regulation boards. Slow controls for status monitoring and state changes are again performed via the specified RS422 line protocol. Each power converter features a trigger input to acquire measurements at dedicated points in time, synchronized within one microsecond. Hence, for slow-controls a fully homogeneous design has been achieved, including the power converters of special magnets.

Injector and synchrotron RF amplifiers feature supplier-provided Siemens PLCs for status monitoring. They interface directly to the SCADA tool at tier-2.

The injector low-level RF system features an embedded, supplier-provided computing device with few modifications to ease pulse-to-pulse modulation operation and cycle-based tagging of measurement data. This Tine-based system is interfaced via the Tine [6] TCP/IP application-level protocol and a custom-built real-time bus for generating stability and duty pulses.

The synchrotron low-level RF system is currently under development at CERN. Its communication interface remains yet to be defined.

Vacuum equipment controllers are self-contained, autonomously operating devices that feature serial interfaces and few digital IO points. They are directly interfaced to the SCADA tool at tier-2 via an OPC server that connects to several terminal servers and a few serial/Ethernet remote IO devices. The same concept applies to the beam interception devices.

Beam diagnostics devices are characterised by a rich set of different front-end electronics and interfaces. A description of the system at tier-4 goes beyond the scope of this article.

Equipment Tier (T-3)

This tier serves two purposes: first, it generates a uniform view of the diverse devices at tier-4 by introducing autonomously working front-end controllers (FEC) that all follow one design pattern. Secondly, the front-end controllers locally keep all configuration data for timely applying configuration changes to achieve efficient pulse-to-pulse modulation operation. Tier-4 devices are virtually represented at this tier either in physical FECs or directly in the SCADA tool at tier-2. FECs are National Instruments PXIe systems running the MS-Windows or Labview-RT operating systems, depending on subsystem specific constraints. FECs are programmed with Labview, using a light-weight layer called Front End Controller Operation System (FECOS) that shields programmers from dealing with communication details, providing design patterns for achieving efficient pulse-to-pulse modulation operation, standard commanding and configuration mechanisms.

Physical FECs communicate with a SCADA system at tier-2 via the OPC protocol. Direct graphical user interfaces for service purposes interface directly to FECs via the NI Shared Variable engine. For high-rate/high-bandwidth measurement data distribution, FECs use an application-level publisher/subscriber protocol based on NI's STM over TCP/IP. FECs may communicate with each other via that protocol to implement non real-time distributed applications or via dedicated low-level signals for real-time synchronization.

Notable physical FECs are the Power Converter Controller (PCC), the REDNet Main Timing System Generator (MTG), the ion source FEC, the injector and synchrotron LLRF FECs to interface to the respective low-level RF control systems.

The main timing system called REDNet [7] distributes events in real-time to synchronize the actions of the FECs. It processes "runs" consisting of lists of pre-defined beam cycles. A beam cycle is represented by a 64bit wide data structure that encodes possible beam characteristics. The MTG acts as single service access point via an application-level TCP/IP protocol for clients that have already successfully allocated a machine partition and that possess a universally unique session identifier (UUID) for this allocation. Before a cycle can be started, the cycle identifier is broadcast to all FECs via the MTG. Once a client has requested the start of a cycle, individual events are broadcast either via a dedicated, optical real-time network and/or the publisher/subscriber mechanism to the frontend applications. Timing system receiver cards generate real-time responses directly to devices via PXI/PXIe real-time bus lines or dedicated hardware interfaces using MRF's Universal IO [8].

The PCC is a PXIe based embedded systems hosting numerous FlexRIO FPGA cards. Each FPGA card can drive up to 6 power converters via an in-house developed, optical real-time link adapter for the NI FlexRIO platform (see Fig. 3). such high-density real-time control of all 260 power converters can be achieved.

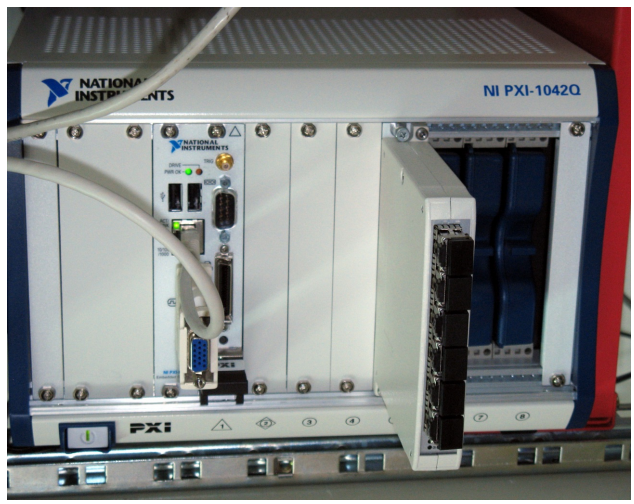


Figure 3: PCC test setup with 1 FlexRIO FPGA board and mounted 6-way optical real-time link adapter.

Finally, this tier hosts also an analogue signals acquisition and distribution system (SADS) that acts as a virtual oscilloscope for up to two concurrent users for about 100 analogue signals. This system is integrated with the MTS so that operators can easily correlate signal over time displays with distributed timing events.

Processing Tier (T-2)

The processing tier (T-2) configures and supervises equipment and frontend tiers. The tier runs supervisory procedures that include machine quality assurance and performance tuning programs. The following list summarizes functions that are located in this tier:

- virtualization of all physical devices as data point structures including a single, uniform state machine for each device that requires state-driven operation,

- creation of a uniform system image by representing all machine partitions, devices, and software components as data points,
- enforcing of role based authentication and access control,
- configuration of front-end controllers via OPC, HTTP and FTP protocols,
- persistent storage of system configuration and device parameters via an RDBMS,
- synchronization and regulation of access to shared resources,
- alarm and error handling, collection of log messages,
- distribution of on-line acquired process data,
- operation data logging and report generation,
- automated execution of supervisory control procedures to acquire machine and beam-related data, to perform quality assurance and performance tuning tasks and to generate new configuration data.

The core system at this tier is a distributed WinCC OA system from ETM. It is the only path to interact with devices during operation periods. Device configuration data may include simple or complex values. They may be static or cycle dependent. A cycle dependent configuration value is assigned a cycle mask and filter, (two 64bit key values). If a software component at FEC level receives the request for a beam cycle, the FECOS framework will look up configuration values based on the cycle mask and filter and provide them to the front-end application. At any time, configuration values can be inspected and modified via WinCC OA. Configuration data are either directly provided to FECs via the OPC protocol or are made available via a dedicated WEB server from which the FECOS executive picks the data.

A particular software component called Virtual Accelerator Allocator (VAA), written entirely in C# is the single entity that accepts requests for machine partition allocations and requests to start a run. This software component interacts solely with the WinCC OA system to achieve its task. Clients issue requests for machine partitions with QoS parameters such as required timing system precision, priority, run mode (medical, medical physics, machine physics or service), name of desired virtual accelerator or working set. If a request cannot be satisfied immediately, the VAA keeps it queued until the client either renews the request or the request times out. All applications in this tier art virtualized using the VMWare server virtualization infrastructure.

Presentation Tier (T-1)

User interfaces are implemented with WinCC OA panels. Enriched user experience such as accelerator synoptics and on-line presentation of measurement data are achieved via re-usable, in-house developed Qt widgets embedded in WinCC OA panels. Data obtained from MAPS can be directly processed these widgets. A panel navigator has been created to allow hierarchical organization of user interface panels, to perform panel-level access control for the defined user roles and to ensure that critical panels cannot be opened multiple times by multiple users at different locations. The CERN/JCOP defined security component [9] is used for implementing fine-grained role based access within panels. A colour and visual language optimized for low-ambient control room use has been devised from best practices in industry and EU norms. A claim mechanism implemented in cooperation between GUI panels and FECOS allows standalone expert user interfaces such as Labview panel applications to be used concurrently with WinCC OA panels to gain direct access to FECs for service purposes without jeopardizing operation safety.

SUMMARY

This article presented the architecture and selected architectural significant components of the MedAustron accelerator control system. This system is based on a 4-tier architecture with the aim to cleanly separate user interfaces, configuration, monitoring and system state processing services, real-time processing and proprietary and third-party supplied accelerator control devices. This control system is an evolution of the CNAO accelerator control system that has recently entered operation. A full vertical column of the system including user interfaces, the main timing system, the configuration repository and the power converter controller has been demonstrated and has proven the validity of the approach. Hence, this architecture may serve as a blueprint for future particle accelerators with comparable operation requirements.

REFERENCES

[1] M. Benedikt and A. Wrulich, “MedAustron – Project overview and status”, Eur. Phys. J. Plus (2011) 126:69

[2] W. W. Eckerson, “Three Tier Client/Server Architecture: Achieving Scalability, Performance and Efficiency in Client Server Applications”, Open Information Systems (1995) 1:10

[3] H. Winick, “Synchrotron Radiation Sources: A Primer”, (1994), pp. 218

[4] L Evans and P Bryant (eds.) “The CERN Large Hadron Collider: Accelerator and Experiments”, JINST (2008) 3, chapter 9, pp.98

[5] H. Pavetits, “Evaluation of the Siemens Safety-Matrix tool with safety PLC for a person and machine protection system of a particle accelerator”, Bachelor thesis FH Wiener Neustadt (2011).

[6] P. Duval and S. Herb, “The TINE Control System Protocol: How to Achieve High Scalability and Performance”, in Proc. PCaPAC 2010, Oct. 5-8 2010.

[7] J. Dedic et al., “Timing System for MedAustron Based on Off-The-Shelf MRF Transport Layer”, in Proc. IPAC 2011.

[8] MRF, Universal IO Specification, last seen at http://www.mrf.fi/index.php/universal-io-modules.

[9] P. Golonka, M. Gonzales-Berges, “Integrated Access Control For PVSS-Based SCADA Systems at CERN”, in Proc. ICALEPCS 2009.

THE RHIC AND RHIC PRE-INJECTORS CONTROLS SYSTEMS: STATUS AND PLANS*

K.A. Brown†, Z. Altinbas, J. Aronson, S. Binello, I. Campbell, M. Costanzo, T. D'Ottavio, W. Eisele, A. Fernando, B. Frak, W. Fu, C. Ho, L. T. Hoff, J. Jamilkowski, P. Kankiya, R. Katz, S.A. Kennell, J. S. Laster, R.C. Lee, G.J. Marr, A. Marusic, R. Michnoff, J. Morris, S. Nemesure, B. Oerter, R.H. Olsen, J. Piacentino, G. Robert-Demolaize, V. Schoefer, R. Schoenfeld, S. Tepikian, C. Theisen, C.M. Zimmer
Collider-Accelerator Department., BNL, Upton, NY

Abstract

For the past twelve years experiments at the Relativistic Heavy Ion Collider (RHIC) have recorded data from collisions of heavy ions and polarized protons, leading to important discoveries in nuclear physics and the spin dynamics of quarks and gluons. BNL is the site of one of the first and still operating alternating gradient synchrotrons, the AGS, which first operated in 1960. The accelerator controls systems for these instruments span multiple generations of technologies. In this report we will describe the current status of the Collider-Accelerator Department controls systems, which are used to control seven different accelerator facilities and multiple science programs (high energy nuclear physics, high energy polarized proton physics, NASA programs, isotope production, and multiple accelerator research and development projects). We will describe the status of current projects, such as the just completed Electron Beam Ion Source (EBIS), our R&D programs in superconducting RF and an Energy Recovery LINAC (ERL), innovations in feedback systems and bunched beam stochastic cooling at RHIC, and plans for future controls system developments.

INTRODUCTION

The BNL Collider Accelerator Department (C-AD) facilities are shown in Fig. 1. The Tandem Van de Graaff (TVDG) accelerators provide heavy ion beams. The 200 MeV LINAC is the source of high-intensity and polarized protons. The Booster synchrotron acts as the second stage of acceleration for beams from both the TVDG and the LINAC. In 2011 EBIS was commissioned and used to provide ion beams to the NASA Space Radiation Laboratory (NSRL) that operates on a beam line off the Booster. Starting in 2012 EBIS will replace the TVDG as the source of ions to RHIC. The AGS acts as a third stage of acceleration before beams are delivered to RHIC. RHIC consists of two accelerators, with counter-rotating beams that can be put into collisions at six intersection points. For normal operations, beams are collided at the STAR and PHENIX experiments. In 2011 a third experiment (AnDY) began tests in the 2 o'clock interaction region. The 4 o'clock region contains the acceleration and storage RF systems for both rings. The 10 o'clock region contains the abort kickers and the beam dumps. The 12 o'clock region contains the polarimeters and hydrogen jet used for polarization measurement and calibration.

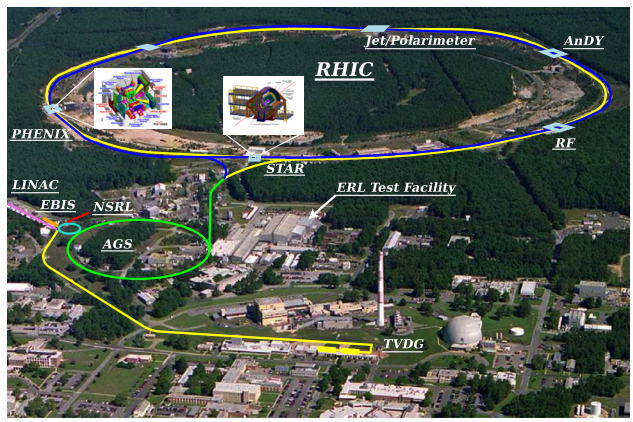


Figure 1: Accelerators at C-AD.

Since the pilot run in 1999 and through to 2011, RHIC has had 6 operating periods (runs) with Au+Au collisions [1, 2, 3]. The Au+Au runs alternated with a d+Au run in 2003 [4], a Cu+Cu run in 2005 [5], and another d+Au run in 2007/08 [6]. We also have run polarized protons, with the first run in 2001. Every year, with the exception of 2007 and 2010 we have had a polarized proton run [7, 8].

RHIC is the first accelerator capable of colliding ions as heavy as gold. As a result, the experiments at RHIC have made significant discoveries, setting the stage for developing a deeper understanding of the quark-gluon plasma (QGP) and the nature of the strong force [9]. The most significant discovery is a new state of hot, dense matter, composed of quarks and gluons, called a "perfect liquid". The properties of this matter can be described by the equations of hydrodynamics when the viscosity is nearly zero. The QGP has been seen to have unusual properties. Symmetry-altering bubbles have been observed, offering hints on the evolution of the early universe. The heaviest antimatter nucleus (an anti-helium-4 nucleus) was discovered at RHIC. RHIC is the world's only accelerator capable of colliding polarized proton beams. The RHIC polarized proton program aims to investigate the missing spin of the proton.

* Work performed under Contract Number DE-AC02-98CH10886 with the auspices of the US Department of Energy.
† kbrown@bnl.gov

In some cases, the spins of the proton's constituent quarks (and antiquarks) only accounts for 30% of its total spin. RHIC experiments are the first to study the spin of gluons and the spin substructure of the proton.

The future of RHIC involves exploring the QGP. RHICs unique capabilities allow a detailed search for the quark matter critical point (the onset of deconfinement), collisions of deuterons or protons on heavy nuclei (so called cold colliding nuclei), collisions of highly deformed nuclei (U+U collisions, so called hot matter) and the exploration of chiral symmetry and thermalization studies. Further into the future is eRHIC, which will collide high energy, high intensity electron beams with ions, polarized protons, and polarized He-3. Such a collider will allow precision imaging of the QGP, determining the spin, flavor, and spatial structure of the nucleon. It will also allow probing more deeply into the nature of the strong force, something still poorly understood, and the properties of gluons, the particles that mediate the strong force.

Controls Systems at C-AD

The C-AD Controls Systems have evolved many times since first installing computer controls in the AGS back in the early 1970's. Today our systems consist of more than 13,000 modules, chassis, fiber optic components, PLCs, and specialty items. Together they combine to provide on the order of ∼1 million setting and measurement control points. The primary Controls hardware platform is the VMEbus chassis, with 315 of these online throughout the facility. Each chassis contains a powerful processor card, and on average contains an additional 10 to 20 custom and commercial VME modules, providing a wide variety of functions and utilities. Some of the types of VME modules include Function Generators, Analog I/O boards, Digital I/O boards, Scalars, Motion Controllers, and specialty Timing boards. A detailed description of the C-AD controls for RHIC is given in ref. [10], although this reference dates back about ten years. Today's system is not much different, although we have many more chassis in the field and have upgraded the performance and bandwidth of our systems. Table 1 shows some of the system parameters for our controls.

The C-AD controls are distributed systems, as shown in Fig. 2. A C++ Accelerator Device Object (ADO) runs on a VME Front-End Computer (FEC) providing access to a power supply or device interface, along with timing and other links (e.g., a real time data link provides basic machine data, such a the main dipole ramp function, directly to FEC level software [11]). Data management consists of a number of servers that together manage data archiving, data logging, process control, and all the various interfaces needed in the main control room (MCR) - alarm managers, controls scripting, system support services, etc.

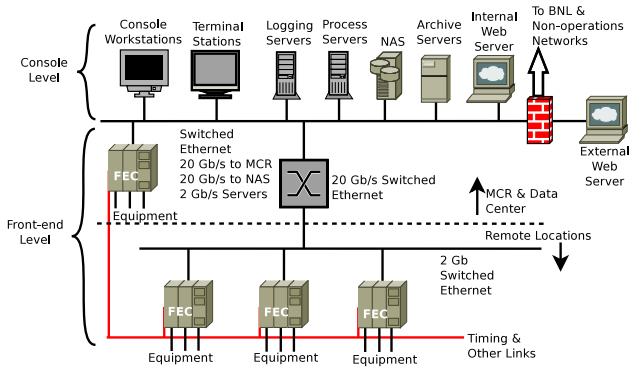


Figure 2: RHIC System Hardware Architecture, 2010.

IMPROVING RHIC LUMINOSITY

Improving the RHIC luminosity is a continuous process. Although RHIC was designed to meet a specific set of parameters, the needs of the experiments are always changing. We now operate RHIC well over an order of magnitude higher than its design. Improvements in the past include methods to make smaller beams through optics manipulations, avoiding processes that tend to increase the beam size, such as intra-beam scattering (IBS), increasing the source intensity, and reducing electron cloud buildup. We also focus on improving integrated luminosity by improving the uptime performance and reducing the time to perform various setup manipulations during the RHIC energy ramp and early part of storing the colliding beams. In the past couple of years we have seen significant increases in the luminosity through the development of bunched beam stochastic cooling. Other improvements include the development of tune, coupling, orbit, and chromaticity feedback [12, 13], correction of 10 Hz orbit oscillations using fast feedback [14], and the development of an all digital low level RF control system in RHIC [15, 16].

EBIS

EBIS consists of a set of external ion sources that inject beam into a solenoid with an ion trap. An electron beam is injected into the solenoid, stripping off electrons bound to the ions. After providing enough time to build up the desired charge states the trap is opened and the ion beam transports to an RFQ and LINAC. The advantages of EBIS include the capability to produce any kind of ion beam (including noble gases and very heavy elements). This will allow RHIC to operate with Uranium beams as well as polarized He-3 [17, 18]. To date, EBIS has reliably provided $He^{+}, He^{2+}, Ne^{5+}, Ne^{8+}, Ar^{11+}, Ti^{18+}$, and Fe^{20+} to NSRL. In 2012 EBIS will provide Uranium and possibly Gold beam to RHIC.

Feedback in RHIC

The development of reliable feedback systems has improved the efficiency of ramp development and the repro-

Table 1: C-AD Controls Systems Parameters, Characterizing the Scale and Performance of our Systems

	2011
Number of front-end VME chassis	315
Number of read/write parameters (settings)	403,000
Number of read-only parameters (measurements)	502,000
Number of name server entries (devices & servers)	52,600
Total archive capacity	75 GB/day
Number of Named Items Archived	260,000

ducibility of the accelerator conditions. The added performance improvement has allowed RHIC to operate under more extreme conditions, helping to increase luminosities and beam polarization. For example, the tighter tune control that is accomplished allows operation close to resonances that would normally destroy the beam. Full descriptions of the various feedback systems can be found in ref.'s [12, 13, 14].

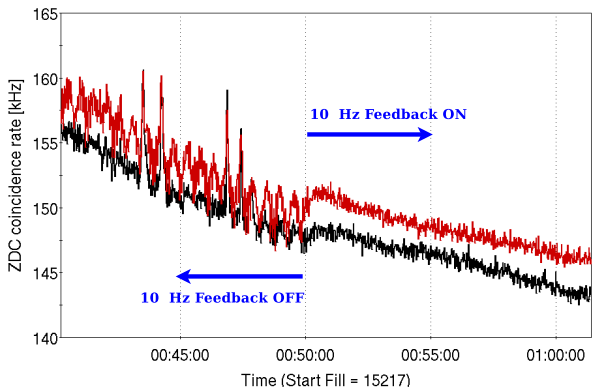


Figure 3: 10 Hz feedback improving the Luminosity at the Experiments. The ZDCs are the zero degree calorimeters used as luminosity monitors.

Stochastic Cooling

The most significant improvement to the heavy ion luminosity in RHIC has been the development of bunched beam stochastic cooling [19]. As can be seen in Fig. 4 the transverse emittance during a store can be reduced, making a smaller beam and maintaining high luminosity during the store. In the 2011 Au+Au run beam was cooled both transversely and longitudinally. This resulted in achieving average store luminosities of $30 \times 10^{26}\ cm^{-2}s^{-1}$ [20], well above the previous record of $20 \times 10^{26}\ cm^{-2}s^{-1}$ from the 2010 run, which included limited cooling, and over a factor of two beyond the $12 \times 10^{26}\ cm^{-2}s^{-1}$, achieved with no cooling in the 2007 run.

Electron Lens

The luminosity of polarized protons in RHIC is limited by head-on beam-beam effects. Beam interactions at the collision points can influence the betatron tunes and tune

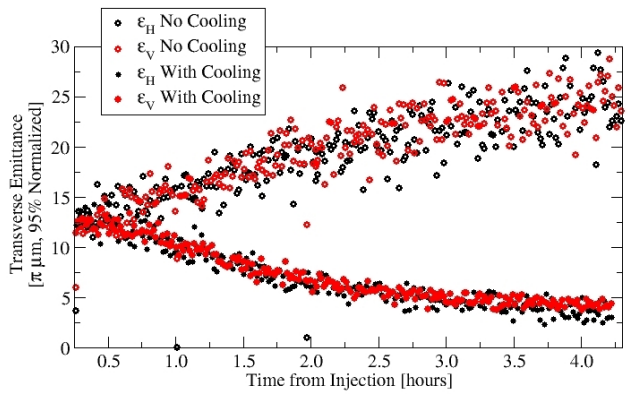


Figure 4: Horizontal and vertical emittances during two different physics stores. The open set of points show the normal increase due to IBS during a store. The solid points show decreasing emittances from transverse cooling. Measured using the RHIC IPM [21]

spreads of the beams. Beam-beam effects can be partially cancelled out by adding a third interaction between the proton beams and an electron beam. A set of electron lenses are being installed in RHIC next year and will be commissioned during the run in 2013. These systems, in combination with an upgrade of the polarized ion source, are expected to improve the polarized proton luminosity by a factor of two [22].

R&D Programs

The C-AD R&D Division was formed at BNL in order to highlight and consolidate the significant accelerator R&D efforts occurring at C-AD. The R&D Division is focused on programs to perform accelerator-based experiments at BNL and to design and construct new facilities.

The main areas of R&D are in superconducting RF systems (SRF), the development of energy recovery LINACs (ERL), the design and construction of eRHIC [23], the development of electron beam sources (high quantum efficiency photocathodes), and to develop coherent electron cooling (CEC) in RHIC [24]. The R&D Division also is closely associated with Stony Brook University's Center for Accelerator Science Education.

The R&D division gets support from the C-AD systems groups, including the Controls Systems groups. The ERL facility is currently being constructed and the controls for

the facility are being developed using the standard C-AD tools and systems (VME based with ADO interfaces).

FUTURE PLANS

Physics Programs

As outlined above, the future of RHIC is the electron-ion collider, adding electron beams, ERLs, and Coherent Electron Cooling to the systems in RHIC. There are three periods of upgrades in the RHIC long term planning. On the short-term (2011-2016) there will be ongoing upgrades to RHIC Luminosity, while the RHIC experiments focus on a well-defined program addressing key open questions in Nuclear Physics, dense matter and the spin structure of the nucleon. On a medium-term (2017-2022), the experiments plan further upgrades to focus on a quantitative pursuit of long-term questions on both cold and hot matter, the strong force, and spin dynamics. On the long-term ($>$ 2022) we enter the eRHIC era adding a 5 GeV (upgradable to 30 GeV) electron accelerator composed of Energy Recovery Linacs inside the RHIC tunnel to facilitate e+Ion, e+p (3He) studies of gluon-dominated cold matter.

Controls Systems

There are many areas in the Controls Systems where we see a need for growth and improvements. These improvements are largely driven by improvements in technologies, but also by improvements in techniques and instrumentation.

Cyber Security is always a concern. The threats are becoming much more sophisticated. New cyber threats reach deeper into the controls hardware with attacks now aimed at PLC and VME level systems. We are actively engaged in instituting measures to protect our systems from these threats.

At C-AD we have a new MCR that is all digital. There are no oscilloscopes or multiplexed signals routed into the new MCR. This puts a greater demand on the software systems. For example, we have built interfaces to remote oscilloscopes that have local signal multiplexing and allow for remote scope control.

Machine protection is a major area of focus. New accelerator systems are coming into operation and new systems are being designed. In addition, there is always a desire to improve reliability and accelerator performance. With these things in mind, we are now working on developing detailed reliability analysis and looking at ways to improve our machine protection systems. We also have new energy sources that can damage components, such as high power synchrotron radiation and high power electron beams.

Accelerator physics online models are improving significantly, which translates into greater computation demands. We also are doing more online analysis (and automatic analysis) which leads to greater data volumes and bandwidth utilization. Techniques such as orbit response matrix measurements (ORM analysis) and online optics measurements are now being done in routine operations, also increasing the computation demands. As we are starting to see today, in the future we expect more physics based controls with parametric interfaces, auto-corrections, and model based corrections. For example, in RHIC, with 10 Hz orbit feedback, we use model derived transfer functions to determine the correct BPM to corrector gains.

More of our systems have all digital controls, such as the new LLRF and new instrumentation systems for feedback. This leads to more information and diagnostics, demanding greater data volumes and bandwidth utilization.

With new feedback systems coming online, we now see new controls paradigms developing. There is an evolution occurring from tuning knobs to adjusting gains, filters, and offsets. Auto-tuning systems are more prevalent as well as automatic parameter corrections. In RHIC we now have the ability to do automatic β^* squeezing during store, while beams are being cooled, to keep the luminosity as high as possible.

REFERENCES

[1] W. Fischer et al., Proc. of EPAC 2004, p. 917

[2] A. Drees et al., Proc. of PAC 2007, p. 722

[3] K. Brown et al., Proc. of IPAC 2010, p. .507

[4] T. Satogata et al., Proc. of PAC 2003, p. 1706

[5] F. Pilat et al., Proc. of PAC 2005, p. 4281

[6] C. Gardner et al., Proc. of EPAC 2008, p. 2548.

[7] M. Bai et al., Proc. of PAC 2005, p. 600

[8] V. Ptitsyn et al., Proc. of EPAC 2006, p. 592

[9] RHIC Science web site, http://www.bnl.gov/rhic/

[10] D. S. Barton et al., Nuclear Instruments and Methods in Physics Research Section A, Vol. 499, 2-3, pp. 356-371, 2003

[11] H. Hartmann, et al., Proc. of PAC 1999, p. 2499

[12] M. Minty et al., Proc. of PAC 2011, p.1395

[13] M. Minty et al., Proc. of IPAC 2011, to be published.

[14] R. J. Michnoff et al., Proc. of PAC 2011, p.609

[15] T. Hayes, K. Smith, Proc. of PAC 2011, p.645

[16] F. Severino, et al., Proc. of PAC 2011, p.672

[17] J. Alessi, et al., Proc. of LINAC 2010, p.1033

[18] J. G. Alessi *et al.*, Rev. Sci. Instrum. **81**, 02A509 (2010).

[19] M. Blaskiewicz, J. M. Brennan and F. Severino, Phys. Rev. Lett. **100**, 174802 (2008).

[20] G. Marr et al., TUPZ038, Proc. of IPAC 2011.

[21] R. Connolly, Presented at the 2010 BIW.

[22] W. Fischer, et al., Proc. of IPAC 2010, p.513

[23] V. N. Litvinenko *et al.*, arXiv:1109.2819 [physics.acc-ph].

[24] V. N. Litvinenko and Y. S. Derbenev, Phys. Rev. Lett. **102**, 114801 (2009).

CONTROL SYSTEM ACHIEVEMENT AT KEKB AND UPGRADE DESIGN FOR SUPERKEKB

K. Furukawa*, A. Akiyama, E. Kadokura, M. Kurashina, K. Mikawa, F. Miyahara,
T.T. Nakamura, J. Odagiri, M. Satoh, T. Suwada, KEK, Tsukuba, Ibaraki, 305-0801, Japan
T. Kudou, S. Kusano, T. Nakamura, K. Yoshii, MELCO SC, Tsukuba Ibaraki, 305-0045, Japan
T. Okazaki, EJIT, Tsuchiura, Ibaraki, 300-0034, Japan

Abstract

SuperKEKB electron-positron asymmetric collider is being constructed after a decade of successful operation at KEKB for B physics research. KEKB completed all of the technical milestones, and had offered important insights into the flavor structure of elementary particles, especially the CP violation. The accelerator control system was led by the combination of scripting languages at the operation layer and EPICS at the equipment layer to a successful performance. The new control system in SuperKEKB will continue to employ those major features of KEKB, with additional technologies for the reliability and flexibility. The major structure will be maintained especially the online linkage to the simulation code and slow controls. However, as the design luminosity is 40-times higher than that of KEKB, several orders of magnitude higher performance will be required at certain area. At the same time more controllers with embedded technology will be installed to meet the limited resources.

INTRODUCTION

KEKB B-Factory was designed as an asymmetric electron/positron collider in order to study the violation of CP symmetry in B-meson system. It consisted of double storage rings of 8-GeV electron (HER) and 3.5-GeV positron (LER) with a diameter of 1 km, and a full energy injector linac of 600 m. It achieved twice as the design luminosity and led to the Nobel Prize of Kobayashi and Maskawa for the theory of quarks and the CP symmetry violation.

After a decade of successful operation at KEKB a new electron/positron collider, SuperKEKB, is being constructed to commission in 2014. It aims at a luminosity of $8 \times 10^{35} cm^{-2} s^{-1}$, 40-times higher than that of KEKB, in order to study the flavor physics of elementary particles further, by mainly squeezing the beams at the collision point. The control system should be utilized to the utmost for the goal.

KEKB was constructed and operated almost the same time as PEP-II at SLAC with the same scientific goal. It provided a friendly competition between these two machines and brought many collaborative efforts. New projects SuperKEKB at KEK and Super B at INFN will be constructed in parallel again. It is hoped to exchange ideas between these projects.

* <kazuro.furukawa@kek.jp>

LESSONS AT KEKB CONTROLS

Success of the high performance operation of KEKB owed much to the control system. It was designed more than 15 years ago and had started the beam operation in 1998. While it inherited a part of resources from the previous project TRISTAN, it restructured most of the software and hardware components. It employed the EPICS (experimental physics and industrial control system) toolkit for the low-level control mechanism and scripting languages for high-level operational applications. The combination provided the flexible and robust operational environment.

Lower-level Controls with EPICS

Before KEKB, projects in the institute repeated the development of its own control system. As technologies such as Unix, VME and TCP/IP became de-facto standards at the end of 1980s, it was considered to share control systems among projects. After the Superconducting Super Collider chose EPICS as the main control toolkit, EPICS became a candidate for future controls at KEK [1]. It was decided to employ EPICS for the KEKB ring controls and the previous software resources were not employed. On the other hand EPICS gateways were constructed for the linac injector to make a bridge from the existent control system [2]. The main reason for that was the ring could shutdown the accelerator completely for 4 years, but the linac had to continue the operation for light sources even during the upgrade construction towards KEKB.

The KEKB ring employed several fieldbuses such as VME, VXI, CAMAC, GPIB and ARCNET depending on the purposes. Approximately 100 VME systems with VxWorks operating system served as EPICS I/O controllers (IOCs) for all the hardware devices including 200 VXI mainframes, 50 CAMAC crates and 200 ARCNET segments [3].

At the linac most of the devices employed IP-network-based controllers. Before the KEKB project network-based PLCs and CAMAC crate controllers, and VMEs were managed by middle-layer software on Unix servers. During the upgrade construction for KEKB, those network-based controllers were shared between old control software and EPICS gateways. Gateways were implemented with portable channel access server (PCAS) at the beginning, and were eventually replaced by soft IOCs as EPICS started to support Unix IOCs [4].

Number of EPICS process variables was approximately 300 thousand, and that of archived ones was 150 thousand. They were distributed over 150 VME-based and Linux-based EPICS IOCs.

High-level Application with Scripting Languages

At the linac Tcl/Tk scripting language was effectively employed for its commissioning [5] after the language had been utilized for testing tools for long time. Later for the both the ring and linac Python was employed as it had more strong points [6]. Many of the device control software were written in those two languages, as well as MEDM.

For the beam operation SAD (Strategic Accelerator Design Program) was extended to have an interpreter, SADscript, which emulated most part of Mathematica language [7]. It provides most of the functionalities which is required by accelerator operation such as linear beam optics, symplectic beam tracking, many of non-linear analysis, optimization, list processing, numerical manipulation, EPICS channel access, and graphical user interface.

During the normal operation it is required to measure the beam response on certain parameter changes, and then to optimize those parameters. Such a process can be interactively carried by SADscript, and then turned into a graphical user interface that is performed routinely. New ideas for luminosity optimization were often proposed in the morning meeting, and corresponding operational tools were realized within a day or two. Some of the ideas turned out to be favorable, and the tool became utilized routinely. As many ideas were proposed, rapid tool development was crucial and SADscript played a significant role. Actually it was difficult to name a single mechanism that enabled the high performance of KEKB, however, for example hundred of rapid one-percent improvements provided twice the performance.

Rejuvenation During a Project

As an upgrade of a control system needs considerable effort, it is preferable to maintain the same environment during a project. However, an accelerator project can span more than ten years, and related software, hardware, persons, companies, and their policies may change substantially during that period. It is necessary to introduce advanced technologies to improve the machine even during the project. On the other hand existent components are often found to be difficult to accommodate new technologies. If a component is modified to accept them, others may have to be modified.

Actually, during the KEKB project new operation schemes were introduced almost every year. As most components in the control system were kept the same, it was rather difficult to catch up with the requirements at the later period. One modification might trigger another, and several modifications had to be performed at the same time. Because the shutdown period in a year is limited, if the extent of modifications exceeded the limit, it became very hard.

Thus, we should be prepared to accommodate small upgrades each year not only for application programs but also for control system infrastructure including base software and hardware components in order to manage a project for a long period.

SUPERKEKB CONTROL SYSTEM

The next project SuperKEKB should follow the existent KEKB control system, which featured with EPICS and scripting languages. It is planned to incorporate new technologies to rejuvenate hardware and software components, which do not change the architecture much. However, two new concepts are considered promising, namely channel access everywhere and dual-layered control system that are described below.

Channel Access Everywhere

The accelerator control architecture in KEK evolved in several steps in the past. Some time ago several control systems were standardized with a combination of several fieldbuses, VME field computers, and Unix computers. In order to consolidate the efforts on the development and maintenance some of the fieldbuses were gradually removed and many controllers were directly attached on to IP networks.

At the same time EPICS control software framework was employed at several control systems at KEK. Eventually, many controllers evolved to embed the same EPICS IOC software as on VME field computers as illustrated in Fig. 1. Common IOC software communicates with others using a common protocol called Channel Access (CA), which realizes unified application development environment from the top to the bottom. We call this embedded EPICS framework as “CA Everywhere”, and it enabled the both rapid development and smooth maintenance [4, 8].

Different kinds of controllers have been developed in the framework of CA Everywhere, and they greatly reduced the control efforts and improved the reliability. They include following controllers. Many of them were re-

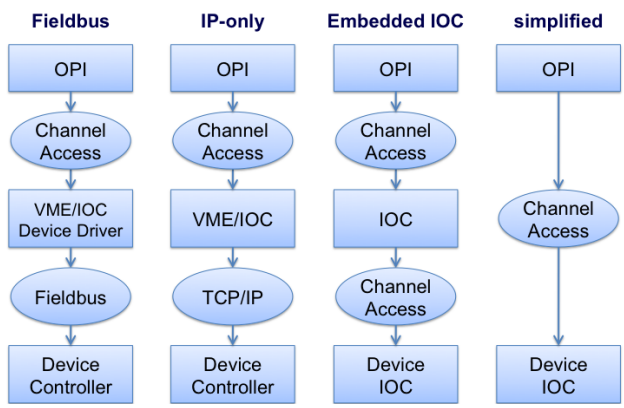


Figure 1: Evolution of device controllers from a fieldbus device towards CA everywhere with embedded IOC.

alized employing FPGA (Field Programmable Gate Array) [9, 10].

- MicroTCA LLRF controller with FPGA and Linux
- Yokogawa FAM3 PLC with Realtime Linux
- Oscilloscope for 50-Hz measurement with Windows
- Timing TDC with Linux
- Microwave power modulator with FPGA and Linux
- Libera BPM readout at 50 Hz with FPGA and Realtime Linux
- NI Compact-RIO with CAS and FPGA

Dual-layered Control System

For higher experimental performance at KEKB and the light sources, it was favorable to inject beams in top-up mode into all the storage rings. In the Photon Factory (PF), a stable beam current would provide precise experimental results. In KEKB, a stable beam current was desired for a sensitive beam collision tuning to increase luminosity.

To that end, simultaneous top-up injection had been established for three storage rings at the KEKB HER and LER, and PF since 2009. Electron and positron beams with very different characteristics, charged from 0.1 nC to 8 nC and with energies of between 2.5 GeV and 8 GeV, were exchanged at a rate of 50 Hz (20 ms). As a result, stored beam stabilities of 0.05 % (1 mA) at the KEKB HER and LER, and 0.01 % (0.05 mA) at PF, were achieved, improving the quality of experiments.

Global and fast controls have been established for such a beam modulation. As the existing control system, based on decade-old hardware and conventional software, was inadequate for controlling the beam within 20 ms, a new event-based control system, capable of regulating over a hundred parameters at 50 Hz, was installed. This system covered the controls of the low-level RF, high-power RF, pulsed magnets, an electron gun, injection systems, and beam instrumentation, whose devices were spread over a 1-km length. While the event-based control system was supervised by the EPICS control software, it had a dedicated communication link for fast, global, and robust controls [11, 12].

An event generator sends timing signals and control data to 17 event receiver stations arranged in a star-like topology. Each link between the event generator and a receiver is carried over a single optical fiber. It provides both synchronized timing signals, with approximately 10 ps precision, and synchronized controls through a realtime software mechanism, at about 10 μs precision. Recent technological advances in FPGA and SFP (Small Form factor Pluggable) enabled reliable controls in this configuration.

The same dual-layered control system with a conventional EPICS control and an event-based control will be essential in SuperKEKB as well. Simultaneous injection will be maintained, as the beam lifetime will be more limited at the SuperKEKB HER and LER. Many more parameters have to be managed precisely in order to realize lower-emittance beams for higher luminosity.

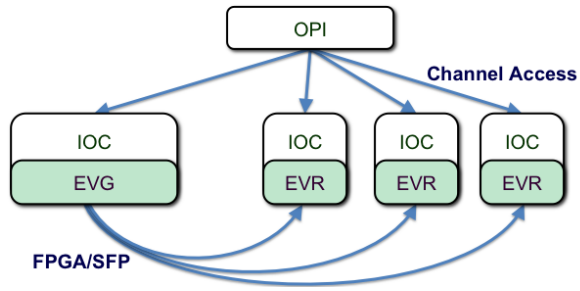


Figure 2: Dual layer controls with EPICS channel access at the top and fast event synchronized control at the bottom.

The event-based control layer manages global, fast controls in the pico- to microsecond range. The EPICS control layer covers slower parameter controls as well as existing conventional controls. Such a layered control system is optimal for the next generation of accelerator systems as depicted in Fig. 2.

CONCLUSION

The design of SuperKEKB controls is being finalized based on the achievement in the KEKB project and the discussion with each sub-group. Control components are constructed preserving EPICS and scripting languages from KEKB and enforcing two additional concepts of "channel access everywhere" and "dual-layer controls".

REFERENCES

[1] EPICS: <http://www.aps.anl.gov/epics/>.

[2] K. Furukawa *et al.*, "Integration Feasibility of the Existing Linac Control System and the Ring EPICS System at KEKB", *Proc. ICALEPCS1995*, Chicago, USA., p. 863.

[3] N. Akasaka *et al.*, "KEKB Accelerator Control System", Nucl. Instrum. Meth. **A 499**, 2003, p. 138.

[4] K. Furukawa *et al.*, "Control System of the KEKB Accelerator Complex", *Proc. ICALEPCS2007*, Knoxville, USA., TOPB04, p. 268.

[5] K. Furukawa *et al.*, "Accelerator Controls in KEKB Linac Commissioning", *Proc. ICALEPCS1999*, Trieste, Italy, p. 98.

[6] T.T. Nakamura *et al.*, "Development of the Software Tools Using Python for EPICS-Based Control System", *Proc. ICALEPCS2007*, Knoxville, USA., TPPA16, p. 120.

[7] SAD: <http://acc-physics.kek.jp/SAD/>.

[8] K. Furukawa *et al.*, "Modern Accelerator Control Systems", *Proc. PAC2007*, Albuquerque, USA., TUZAC02, p. 873.

[9] M. Satoh *et al.*, "The BPM DAQ System Upgrade for SuperKEKB Injector Linac", in these proceedings.

[10] K. Furukawa *et al.*, "Embedded LLRF Controller with Channel Access on MicroTCA Backplane Interconnect", ibid.

[11] K. Furukawa *et al.*, "New Event-based Control System for Simultaneous Top-up Operation at KEKB and PF", *Proc. ICALEPCS2009*, Kobe, Japan, THP052, p. 765.

[12] K. Furukawa *et al.*, "Pulse-to-pulse Beam Modulation and Event-based Beam Feedback Systems at KEKB Linac", *Proc. IPAC2010*, Kyoto, Japan, TUOCMH01, p. 1271.

THE LHC EXPERIMENT CONTROL SYSTEM: ON THE PATH TO FULL AUTOMATION

C. Gaspar, F. Alessio, L. Cardoso, M. Frank, J.C. Garnier, E. v. Herwijnen, R. Jacobsson, B. Jost, N. Neufeld, R. Schwemmer, CERN, Geneva, Switzerland
O. Callot, LAL, IN2P3/CNRS and Université Paris 11, Orsay, France
B. Franek, Rutherford Appleton Laboratory, Chilton, Didcot, OX11 0QX, UK

Abstract

LHCb is a large experiment at the LHC accelerator. The experiment control system is in charge of the configuration, control and monitoring of the different sub-detectors and of all areas of the online system. The building blocks of the control system are based on the PVSS SCADA System complemented by a control Framework developed in common for the 4 LHC experiments. This framework includes an "expert system" like tool called SMI++ which is used for the system automation. The experiment's operations are now almost completely automated, driven by a top-level object called Big-Brother, which pilots all the experiment's standard procedures and the most common error-recovery procedures. The architecture, tools and mechanisms used for the implementation as well as some operational examples will be described.

INTRODUCTION

LHCb [1] is one of the four experiments at the Large Hadron Collider (LHC) at CERN. LHCb's Experiment Control System (ECS) handles the configuration, monitoring and operation of all experimental equipment in all areas of the Online System:

- The Data Acquisition System (DAQ): front-end electronics, readout network, storage etc.
- The Timing and Fast Control System (TFC): timing and trigger distribution electronics.
- The L0 Trigger (L0): the hardware trigger components.
- The High Level Trigger (HLT) Farm: thousands of trigger algorithms running on a large CPU farm.
- The Monitoring Farm: A smaller farm running monitoring tasks to produce histograms for checking online the quality of the data being acquired
- The Detector Control System (DCS): gases, high voltages, low voltages, temperatures, etc
- The Experiment's Infrastructure: magnet, cooling, electricity distribution, detector safety, etc
- Interaction with the outside world: LHC Accelerator, CERN safety system, CERN technical services, etc.

The relationship between the ECS and the other online components of the experiment is shown schematically in Fig. 1. This figure shows that the ECS provides the unique interface between the users and all experimental equipment.

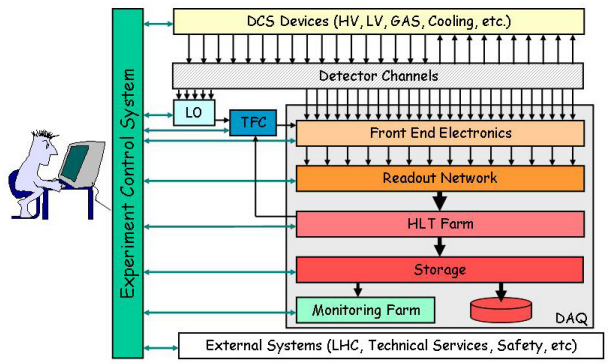


Figure 1: Scope of the Experiment Control System.

In order to achieve an integrated and coherent control system throughout all areas of the online system, a common approach was taken in the design of the complete system and the same tools and components were used for the implementation of the various parts of the system.

A common project, the Joint Controls Project (JCOP) [2], was setup between the four LHC experiments and a Controls group at CERN, to define a common architecture and a framework to be used by the experiments in order to build their detector control systems. LHCb extended the concept, and used these tools for the implementation not only of the DCS, but of all areas of control in the experiment.

SYSTEM ARCHITECTURE

The size and complexity of the LHC experiment's control systems have driven the choice of the system's architecture.

JCOP adopted a hierarchical, highly distributed, tree-like, structure to represent the structure of sub-detectors, sub-systems and hardware components. This hierarchy allows a high degree of independence between components, for concurrent use during integration, test or calibration phases, but it also allows integrated control, both automated and user-driven, during physics data-taking.

The building blocks of this tree can be of two types: "Device Units", the tree leaves, which are capable of "driving" the equipment to which they correspond and "Control Units" (CUs) which correspond to logical sub-systems and can monitor and control the sub-tree below them. Figure 2 shows a simplified version of LHCb's control system architecture.

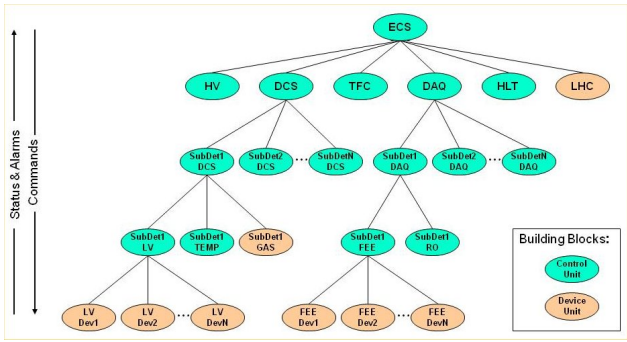


Figure 2: LHCb Simplified Architecture.

THE FRAMEWORK

The JCOP Framework [3] provides for the integration of the various components (devices) in a coherent and uniform manner. JCOP defines the framework as:
"An integrated set of guidelines and software tools used by detector developers to realize their specific control system application. The framework will include, as far as possible all templates, standard elements and functions required to achieve a homogeneous control system and to reduce the development effort as much as possible for the developers".

The JCOP Framework was implemented based on a SCADA (Supervisory Control And Data Acquisition) system called PVSSII [4]. While PVSSII offers most of the needed features to implement a large control system, the "Control Units" described above are abstract objects and are better implemented using a modeling tool. For this purpose SMI++ [5] was integrated into the framework.

SMI++ is a toolkit for designing and implementing distributed control systems, its methodology combines three concepts: object orientation, Finite State Machines (FSM) and rule-based reasoning.

The framework offers tools to implement a hierarchical control system, as described in the architecture chapter, in particular a graphical user interface, shown in Fig. 3, which allows the configuration of object types, declaration of states, actions, rules, etc. as well as the definition and operation of the hierarchical control tree composed of the two types of nodes below.

Device Units

Device Units provide access to "real" devices. In LHCb a device is basically any hardware or software entity that needs to be controlled and/or monitored, it can range from a simple temperature probe to a very complex electronics board or to a trigger process in a large HLT farm. Device Units are mostly implemented using standard PVSS tools.

PVSS provides drivers for different types of commercial hardware and a publish/subscribe protocol (DIM [6]) was interfaced to PVSS to access any non-commercial hardware or software devices. The interface to a device unit is defined as a Finite State Machine. I.e. a device is always in a well-defined state and can receive commands depending on its state.

Control Units

Control Units are logical sub-systems and perform as local decision units. They can take decisions and act on their children, i.e. send them commands, based on their states. Any Control Unit and the associated sub-tree can be a self-contained entity. The logical behavior of a Control Unit is expressed in terms of Finite State Machines. State transitions can be triggered by command reception, either from its parent or from an operator or by State changes of its children. State transitions cause the evaluation of logical conditions (rules) and possibly commands to be sent to the children. This mechanism can be used to propagate actions down the tree, to automate operations and to recover from error situations. The behavior of the Control Units is described and implemented using the SML language which is part of the SMI++ toolkit.

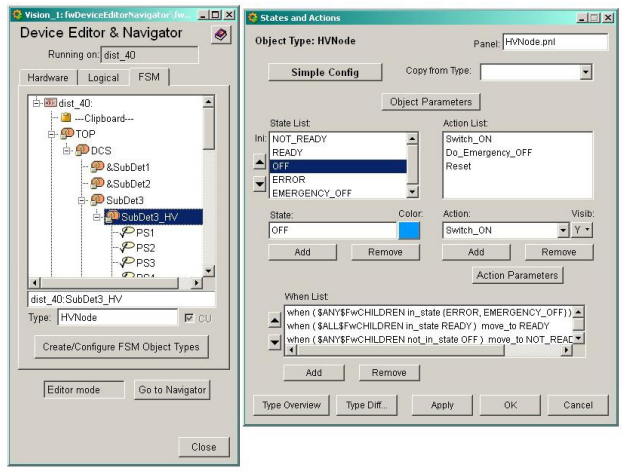


Figure 3: The Framework Device Editor Navigator.

GUIDELINES & TEMPLATES

LHCb is composed of hundreds of sub-systems provided by many different teams from institutes all over the world. The JCOP Framework was distributed to all these teams in order to implement their specific control systems. But in order to make sure to achieve a coherent control system some further guidelines were also specified.

There are various types of equipment being controlled in the various sub-systems, in particular they are normally operated at different times. For example the gas systems should be stable throughout a complete running period, while the high voltages may need to be switched off when the accelerator injects beam. In order to be able to operate all equipment in the correct order three Control Domains have been defined:

- DCS - For the equipment whose operation and stability is normally related to a complete running period. Example: Gas, Cooling, Low Voltages, etc.

- HV - For the equipment whose operation is normally related to the machine state. Example: High Voltages
- DAQ - For the equipment whose operation is related to a "Run". Example: Readout electronics, High Level Trigger processes, etc.

Due to their different characteristics the equipment in the various domains has different states and accepts different actions. For each domain a template implementation of the corresponding Finite State Machine was developed and distributed as a framework component to all sub-detectors. Figure 4 shows the FSM implemented for each domain.

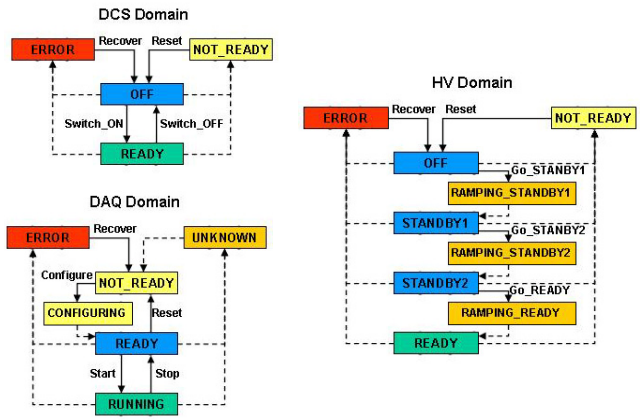


Figure 4: LHCb Finite State Machine Templates

ECS & AUTOMATION

Using the templates described above, the control tree shown in Figure 2 was implemented. The top-level "ECS" node integrates all underlying sub-systems. This node, is, in fact decomposed in three main objects, represented in Fig. 5.

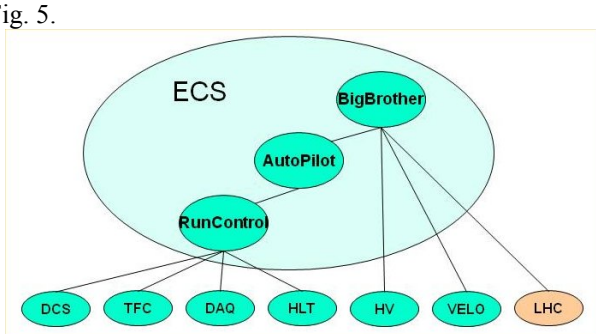


Figure 5: ECS main components

In order to prevent human mistakes and to speed up standard procedures, the system should be, as much as possible, fully automated. Since the same framework, and the same templates are used throughout all sub-systems, the implementation of automation rules within or across sub-systems was a very simple task. Some examples of automated procedures are described below.

High Level Trigger Control

The HLT is performed by a farm of around 1500 PCs, each one running several instances of Trigger processes, in total around 40000 processes are monitored and controlled by the HLT control sub-system.

In such a large system it can happen that not all PCs are operational at a given point, the control system has implemented mechanism to:

1. Automatically exclude misbehaving PCs, either because they take too long to configure, or because they stop responding. If a certain PC misbehaves several times in a row it gets marked as "bad".
2. Consider the farm ready to start a run when a certain percentage of the farm (70% at the moment) is ready, in order to speed up the start-of-run procedure.
3. Once the run is going try to include back the excluded PCs (if not marked as bad). Any PC included at run time, will go automatically through all steps (Configure, Start, etc.) until it is in RUNNING state and processing events like all others.

In general any PC or group of PCs can be transparently excluded or included at run time.

The Run Control

The Run Control provides the main user interface to the Data Acquisition system, it allows operators to include/exclude sub-systems or sub-detectors, to define the current Activity (used in order to configure all sub-systems) and to stop and start runs whenever required. But it also implements some automated actions. In particular it detects problems during the run, for example if sub-detectors are desynchronized, it can by itself issue a "CHANGE_RUN" command, which re-synchronizes all sub-detectors or it can reset and reconfigure the sub-detector responsible for the problem.

These problems are detected by the Run Control itself by checking his own counters, like dead-time or trigger rates, but some problems can only be detected by analysing the event data being acquired. The monitoring tasks while checking the data (since they are integrated in the control system like any other device) also provide flags that can instruct the Run Control to issue automatically a run change or reset a particular sub-detector.

The Auto Pilot

The Auto Pilot's responsibility is to keep the system running. It knows how to sequence operations to get LHCb, and all its sub-systems, running from any state.

Once the system is running, if there is any problem, it will try to recover and get the system back running. Of course it will only try a certain amount of times (5 at the moment), if it doesn't manage it stops and asks for help from the operator.

Big Brother

The, so called, Big Brother control object handles the dependencies between LHCb and the LHC Accelerator. For example, when the accelerator injects beams the sub-detector high-voltages must be kept in a safe state.

Another specificity of LHCb is the Vertex Locator (VELO) sub-detector, which physically moves closer to the beam when the accelerator declares stable beams.

Big Brother's operations are driven by the LHC state, whenever the LHC changes state the appropriate action is sent to the sub-detector's high and low voltage sub-systems, to the VELO and/or to the RunControl through the AutoPilot. For example the sequence of operations when the LHC moves to state "PHYSICS" is the following:

1. Send "Goto_PHYSICS" to all sub-detector's voltages (and in particular the VELO HV)
2. When the beam position is received from a VELO Monitoring task -> Start Closing the VELO
3. When the VELO is closed (centred around the beam position) -> Send a "CHANGE_RUN" to the RunControl in order to cleanly mark the start of physics data taking.

At the moment most LHC State changes need to be confirmed by the operator before the associated actions are sent out to the LHCb sub-systems.

SIZE AND PERFORMANCE

The complete control system, now in production, runs distributed over around 150 PCs, where around 100 are Linux machines and around 50 run Windows.

The Linux machines are mostly used for the control of the HLT farm (50 control PCs) and the data acquisition systems of the sub detectors. While the Windows machines are mostly used for the detector control systems. The whole system is composed of around 2000 control units and more than 50000 device units.

In order to give an idea of the performance of the system: a cold start of the data acquisition system, i.e. the time to completely configure the system and start a run is around four minutes (this includes configuring all sub-detector electronics and starting and configuring around 40000 trigger processes).

In practice a cold start is rarely performed since the run is normally started well before the LHC declares "Stable Beams" and we have implemented a "Fast Run Change" mechanism which allows to stop/start a run in a few seconds (around 5 seconds) whenever the run conditions change, for example when the VELO has moved to nominal position at start of fill, or in order to change trigger settings.

Recently, in particular after the latest addition of automated operations, the Data Acquisition Efficiency is normally around 98 %.

LHCB OPERATIONS

LHCb is now running routinely. For the complete operation of the experiment 2 operators are permanently on shift in the control room: the "Data Manager" responsible for checking the data quality and the "Shift Leader". The Shift Leader's main task is the supervision of the experiment's state and of the data taking activities. For this task he/she uses mainly the Run Control user interface, and the Big Brother user interface, both shown in Fig. 6.

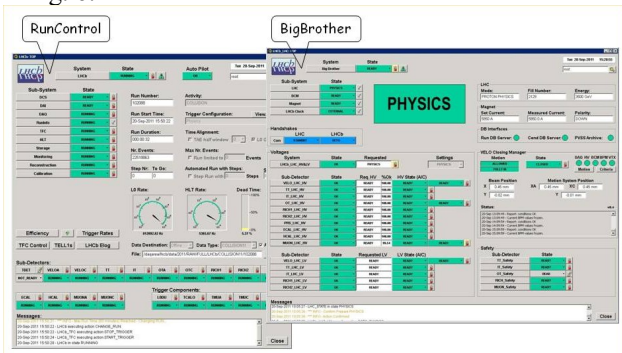


Figure 6: Shift Leader main User Interfaces.

Experience with system operation is very positive, after a short training course any member of the experiment, without previous online experience, is capable of piloting the system.

CONCLUSIONS

LHCb has designed and implemented a coherent and homogeneous control system. The Experiment Control System provides a complete, summarized view of the experiment. It allows to configure, monitor and operate the full experiment either in an integrated, global way for normal physics data taking or to run any combination of sub-detectors in parallel and in standalone.

The system is now being used daily for Physics data taking and for all other global or stand-alone sub-detector activities with very positive results.

Some of the most important features of the ECS, such as the sequencing of operations and the automation and error recovery mechanisms, come from the integration of the SMI++ toolkit within the JCOP PVSS based framework.

LHCb operations are now almost completely automated, which makes the operator task much easier and allowed to greatly improve the overall system efficiency.

REFERENCES

[1] LHCb Collaboration, "LHCb – the Large Hadron Collider beauty experiment, reoptimised detector design and performance", CERN/LHCC 2003-030

[2] D.R. Myers et al, "The LHC experiments Joint COntrols Project, JCOP", Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (Trieste) Italy, 1999

[3] S. Schmeling et al, "Controls Framework for LHC experiments" Proc. 13th IEEE-NPSS Real Time Conference (Montreal) Canada, 2003

[4] PVSS-II http://www.pvss.com

[5] B. Franek and C. Gaspar, "SMI++ - an object oriented Framework for designing distributed control systems", IEEE Trans. Nucl. Sci. **45** 4 1946-50, 1998.

[6] C. Gaspar, M. Dönszelmann and Ph. Charpentier, "DIM, a portable, light weight package for information publishing, data transfer and inter-process communication", Computer Physics Communications **140** 1+2 102-9, 2001.

MANAGING MAYHEM*

K. S. White, ORNL, Oak Ridge, TN, 37831, U.S.A

Abstract

Research institutes, typically full of excellent scientists and engineers, tend to be very focused on the technical aspects of their work, but reluctant to put much energy into the management functions that enable a healthy, productive organization. This is not really surprising when one considers that scientists and engineers are well trained to measure and evaluate quantitative entities while the management arena is dominated by qualitative concepts. Management is generally considered to include planning, organizing, leading and controlling. This paper discusses the essential management functions and techniques that can be employed to maximize success in a research and development (R&D) organization.

MOTIVATION

Roughly half of all scientific R&D is underwritten by public funding. With ever increasing pressure on governments to reduce spending, it is more important than ever to effectively manage these projects and demonstrate the long term value to the public. For R&D undertaken by private companies, there is even more pressure to demonstrate the investment will lead to profitable products sometimes with very short timelines.

In contrast to most for-profit companies, where goods or services are produced and success is measured almost exclusively by revenue, R&D organizations deal in innovation and measure success in terms knowledge gained. Of course dollars are easy to count but advances in understanding elude quantification.

By its very nature, R&D requires trial and error, creativity, and even failures that lead to rework, to move towards discovery. These critical ingredients are in fact the same variable factors that traditional management techniques strive to eliminate from processes for the sake of consistency and efficiency. The challenge for managers of R&D is how to enable the mayhem needed to fuel innovation while simultaneous assuring the needed progress to keep sponsors satisfied and therefore support future funding.

BACKGROUND

Modern management methods can be traced Frederick Taylor, a mechanical engineer who first published his theories in "The Principles of Scientific Management" in 1911 [1]. It was Taylor's work that first introduced the concept of management as a distinct profession. His work was primarily aimed at the manufacturing industry, which drove the economy during the Industrial Age. Taylor sought to maximize productivity and advocated careful (scientific) study and measurement of each step of the work process to determine the most efficient method. Trained workers should execute these steps without deviation and be paid more for achieving higher productivity.

Taylor's methods proved effective and were widely adopted, laying the groundwork for industrial engineering and quality control. This autocratic approach, with a strong emphasis on the planning, organizing and controlling functions of management, did have a positive impact on productivity and Taylor's influence is still very present in management training and practices today.

Conversely, Taylor's techniques essentially viewed the workers as interchangeable parts of a machine, with pay as their only reward. They had no input into the prescribed processes and were likely to be assigned to perform the same limited function for long periods of time with no consideration of job satisfaction or avenues for skills development. This naturally led to boredom and job dissatisfaction and worker potential remained largely untapped in this system.

Subsequent management studies and theories such as Abraham Maslow's "Hierarchy of Needs", Elton Mayo's "Hawthorne Studies", Frederick Herzberg's "Hygiene and Motivational Factors" and McGregor's "Theory X and Theory Y" introduced elements of human behavior and employee motivation as significant contributors to overall productivity [2].

For R&D managers, one of the most relevant contributors to modern management theory was Peter Drucker who first introduced the term "knowledge worker" in 1957 in his book "The Landmarks of Tomorrow" [3], signaling the future shift from the Industrial Age to the Information Age. In many subsequent publications, Drucker elaborated on how to improve productivity in knowledge work, advocating a high degree of self-management for knowledge workers and viewing knowledge workers as assets rather than costs [4].

WHY IS MANAGEMENT DIFFERENT FOR R&D ORGANIZATIONS?

The primary difference between managing commercial activities and research lies in the inherent uncertainty of the work. By its very nature, research is unpredictable, making it difficult to define desired outcomes, much less forecast each needed step along with associated timelines and costs. Research is nurtured by experimentation and missteps, which are largely excluded by popular management methods, but are vital to discovery.

Another important difference between for-profit companies and scientific entities lies in the typical characteristics of the workforce. Professionals employed by R&D organizations tend to have a higher potential and level of education than the average worker. The best R&D

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy

professionals work well independently and have a strong need for autonomy.

Management of R&D organizations is also complicated by the tendency to place people in management roles primarily because they have demonstrated technical excellence as a scientist or engineer. In this system, a high performing technical person is assigned a management role without any preparation or consideration of characteristics desirable in good manager. This even occurs when the incumbent would prefer to spend the bulk of their time pursing their technical work, but accepts the management role because it is viewed as a way to progress on the career ladder. In this case, it is not uncommon for the newly appointed manager to continue pursuing their research and largely ignore their management duties.

MANAGEMENT FUNCTIONS

The traditional management functions - organizing, planning, controlling and leading - can be overly prescriptive when applied to unpredictable R&D work and demotivating to knowledge workers. Without adjustments for the nature of the work, managers can spend a lot of time on functions that do not help productivity or help the organization meet its goals.

Organizing

The organizing function involves developing a structure and allocating resources to achieve objectives. This implies understanding the required skills mix and recruiting, training, developing and retaining people with the appropriate skills. The organization of a group should take into account how people will be deployed. For example, a group may be organized around projects and this works in the simplest case where people work on only one project at a time. In reality, people are often assigned to more than one project at a time. This leads to the need for more complex structures where people are grouped by skills or functions, and then divide their time between various projects. Given the dynamic nature of R&D, managers should consider organizing in ways that can easily adapt to changing project needs and ensure people can work across organizational boundaries as needed to enable success. The organization structure ultimately influences how people interact and share information.

Planning and Controlling

There are many types of planning, but two important types are strategic planning and work planning. Theoretically, strategic planning begins at the highest level and sets the overall vision and goals of an organization. From strategic plans, sub-organizations set their goals which eventually are divided into group goals and then projects. Work planning typically involves breaking a project down into a series of scheduled tasks along with cost estimates. A typical project plan includes a long list of tasks organized in a timeline along with the resources needed, both labor and materials, to accomplish each task. These plans form a baseline used periodically to assess progress and determine success.

When guided only by traditional management theories and techniques, R&D managers struggle with planning due to limited precedents for their work, the unpredictability of the R&D process and the inability to form clear definitions of successful outcomes. This makes it challenging to generate accurate plans and cost estimates.

Customary management dictates that managers control work by periodically checking project progress versus established goals and objectives detailed in work plans. When progress does not meet expectations, managers are expected to take corrective action to get things back on track.

Too much emphasis on planning and controlling R&D work can lead to great frustration and imply a lack of success when in fact good progress is being made. R&D projects could be better served by reducing the level of detail in plans to reflect the uncertainty of the project. Rather than scheduling strict milestones, managers could define logical points when decisions should be taken about the future path of the research. Plans should build in flexibility and be viewed as a living documents rather than a strict baselines.

Without highly predictable detailed plans, the idea of controlling a project is really futile. For R&D, it is necessary to evaluate progress, but this should be primarily for the purpose of adjusting the plan according to lessons learned. Rather than controlling, an R&D manager should focus on guiding or steering the process in collaboration with key contributors.

Leading

Leadership is perhaps the most important function for R&D managers. First and foremost, leaders establish and communicate their vision for the organization. Articulating an inclusive vision serves to inspire employees and energize them towards accomplishing organizational goals. Leadership also includes establishing an environment that enables success. R&D efforts thrive where people are free to try new methods and openly disseminate the results, even when unsuccessful. Such an environment is key to realizing real innovations, as opposed to the small incremental improvements that otherwise occur.

There are many different leadership styles, often defined by the approach a leader takes to decision making. Because R&D professionals are generally intelligent, educated and value autonomy, effective R&D managers can benefit from engaging key contributors in the decision making process. This is known as the “participative leadership style” and enables better decisions by tapping into the collective intelligence of the organization [5].

OTHER CONSIDERATIONS

Communication

The best managers are good communicators. Many people confuse communication with talking. The best leaders listen more than they talk and engage in active listening, and asking well-placed questions. Good listening not only has the benefit of providing the manager with information, it sends a message to employees saying the manager values their work.

On the other hand, when managers have a message to disseminate, it is useful to remember what marketing experts refer to as the "Rule of Seven". This rule says that people need to hear or see a message at least seven times, preferably in multiple formats, before they will act on it. With the large volume of information that bombards employees today, it makes sense to reinforce your message until it becomes part of the collective consciousness.

Motivation

People generally perform best when given the opportunity to do things they are good at and enjoy. Matching projects to an individual's expertise and interests helps to keep people engaged and interested in their work. Particularly in the world of knowledge workers, giving people a high level of autonomy is considered necessary to fully realize their potential.

Higher productivity can often be achieved by letting capable technical professionals "self-manage" their work to a much higher degree than would be prescribed by traditional management methods. Some studies indicate that productivity can be increased by a factor of two if managers adopt an untraditional leadership style that puts the focus on the ideas of others rather than their own. This strategy is known as the "The Multiplier Effect" and challenges the traditional role of managers as decision makers and advocates using the intelligence of everyone to stimulate the flow of ideas and healthy technical debate that lead to real innovation [6].

In this type of approach, the manager must create an environment of open communication and actively encourage other people to bring forward their ideas. Furthermore, the manager has to adopt a role that is more about facilitating and enabling others than controlling everything personally. This might make some managers insecure about their own value to the organization, However, if one considers the possibilities of tapping more deeply into the intelligence and potential of everyone on staff, not just managers, for good ideas it is easy to see that we might be sitting on a great deal of latent value.

Teamwork

While not every activity requires teamwork, little R&D of any importance can be accomplished by the effort of a single individual working in isolation. Unfortunately, many managers lack the skills to build effective teams. Teamwork depends on a foundation of trust, the ability to openly discuss ideas and commit to a course of action and members who hold each other accountable for results. Patrick Lencioni has written extensively on how to create great teams and effectively lead for success [7].

Conflict

If everyone is encouraged to openly discuss ideas, there is bound to be some disagreement. Managers often spend considerable effort suppressing conflict only to create a superficial outward harmony amongst the staff. When conflict is suppressed, it simply moves underground where it becomes more harmful. It is far better to create a safe environment where all inputs are valued and different viewpoints can be openly discussed. From this type of healthy debate, the best ideas will be strengthened and valuable information emerges leading to better technical decisions. The creativity that leads to innovation is difficult to cultivate without the open discussion of ideas. Rather than smoothing over differences, managers are better served spending their effort to teach employees how to discuss ideas without interpreting critiques as personal affronts.

SUMMARY

Successful management for production oriented businesses tends to rely heavily on the planning, organizing and controlling functions. To be successful in an R&D organization, the manager must be willing to relinquish some control and focus more effort on leading and coordinating the abilities of highly intelligent, educated and autonomous individuals. Good managers invest heavily in developing talented people and creating an environment that lets people do their best work

REFERENCES

[1] R. Kanigel, "The One Best Way: Frederick Winslow Taylor and the Enigma of Efficiency", Viking Penguin, 1997.
[2] http://en.wikipedia.org/wiki/Motivation.
[3] P. Drucker, "The Landmarks of Tomorrow", Harper, New York, 1957.
[4] http://www.knowledgeworkerperformance.com /Peter-Drucker-Knowledge-Worker-Productivity.aspx.
[5] R. K. Jain and H. C. Triandis, "Management of Research and Development Organizations", John Wiley & Sons, Inc., New York, 1997.
[6] L. Wiseman and G. McKeown, "Multipliers: How the Best Leader Make Everyone Smarter", HarperCollins, New York, 2010.
[7] P. Lencioni, "The Five Dysfunctions of a Team", Jossey-Bass, San Francisco, 2002.

MANAGING THE DEVELOPMENT OF PLANT SUBSYSTEMS FOR A LARGE INTERNATIONAL PROJECT

Dave Gurd, Vancouver, Canada

Abstract

ITER is an international collaborative project under development by nations representing over one half of the world's population. Major components will be supplied by "Domestic Agencies" representing the various participating countries. While the supervisory control system, known as "CODAC", will be developed "in fund" by the International Organization at the project site in the south of France, the EPICS and PLC-based plant control subsystems are to be developed and tested locally, where the subsystems themselves are being built. This is similar to the model used for the development of the Spallation Neutron Source, which was a US national collaboration. However the much more complex constraints of an international collaboration preclude the use of many specifics of the SNS collaboration approach. Moreover, procedures for final system integration and commissioning at ITER are not yet well defined. This paper will outline the particular issues either inherent in an international collaboration or specific to ITER, and will suggest approaches to mitigate those problems with the goal of assuring a successful and timely integration and commissioning phase.

CONSTRAINTS AND CONSEQUENCES

ITER is too big and expensive an undertaking not to be done as an international collaboration. Without such a collaboration, bringing together the technical and financial resources of most of the developed world, ITER would simply not happen. Members of the partnership have different cultures, different scientific governance, different funding mechanisms, and possibly even different reasons for participating, with the result that political imperatives must inevitably result in some compromises and extra overhead. In addition to addressing very difficult technical challenges, ITER must learn, perhaps often only by trial and error, how best to minimize those inefficiencies and live with certain constraints. With success, ITER should become a model for future large international collaborations, such as the ILC.

As much as possible of the work of building ITER must be shared among the participating countries through their Domestic Agencies (DAs). This was the *sine qua non* of international participation, and it constrains the amount and type of work that is done by the ITER Organization (IO) at Cadarache. The work of IO is primarily to prepare contracts with the DAs, the specifications for these contracts sometimes being "build to print" but more commonly functional specifications. Very little "hands-on" work is done at IO, and much of the design work is accomplished by external contractors.

CODAC, the central common high-level supervisory control system for ITER, is developed "in fund" by IO as opposed to "in kind" by the DAs. The same approach is used for CODAC development as for other IO work, and many of the same constraints apply. Some of the constraints that affect the work and management of the CODAC group are discussed in the paragraphs below.

Staff Size Limitation

In part for reasons of cost containment and in part because of the ITER emphasis on outsourcing, the size of the permanent CODAC staff is severely constrained. The total manpower effort for CODAC design and development at Cadarache is currently set at a level comparable to that used for SNS – a project of about one third the scope. As a consequence, CODAC is required to outsource most of its development work.

Outsourcing

The R&D program has been carried out largely by external contracts, as is (and will be) much of the development of ITER-specific hardware and software. The technical staff is therefore required to spend much of its time writing technical specifications and managing contractors – a task that does not make optimal use of their training and skills. Contractors are located in partner countries all over the world and supervision is impaired by the resulting lack of face-to-face meetings. Although the practice is discouraged by senior management (contracts with well-defined deliverables are preferred), there has been some success with "insourcing," defined here as supplementing staff with external contractors working on site on relatively loose "framework" contracts. More recently, limits have been set on external contractor remuneration that are in some cases below market rates. As a result, some contracts have been suspended, and others cannot be awarded.

Laboratories

For reasons having to do partly with (possibly temporary) space limitations, partly with safety considerations, and partly deriving from the general approach of limited "hands-on" work by IO, the CODAC team currently has only a small "technical area" for prototype testing and development – too small even for a technician's bench. Software developers seldom have the opportunity to turn on a light they can see. As a consequence, they are somewhat disconnected from the concept that what they are doing must eventually relate to real equipment.

Plant System Integration

The most important constraint on the work of the CODAC Group deriving from the ITER agreement is the nature of the interface with the various plant systems – the major components of ITER. It is the role of the ITER Controls Group and CODAC to effect an integration of these components in such a way as to provide to operations a uniform view of an extremely complex system. It is this integration that presents the major challenge for the CODAC Team.

THE INTEGRATION CHALLENGE

The ITER project is broken down into 33 "plant systems." (Magnets, Cryogenics, Cooling Water, Buildings, etc.) These "plants" are to be delivered to the ITER site at Cadarache "in kind" by the participating Domestic Agencies. In many cases, however, the plants are made up of a number of subsystems each with its own specific functional requirements and each to be delivered separately *with its own Instrumentation and Control (I&C) system* by the responsible Domestic Agency. These "stand-alone" I&C systems are referred to as "Plant System I&Cs." There are approximately 220 such Plant I&Cs. (The exact number is a function of design and is subject to change.) All of these plants, plant subsystems and plant system I&Cs are the subjects of contractual arrangements between ITER and the responsible DAs. In some cases more than one DA is responsible for the delivery of a single plant subsystem. In those cases, responsibility for testing and integration remains to be clarified. In any case, it is the responsibility of the CODAC group to do the final integration of these 220 plant system I&Cs with the CODAC infrastructure at Cadarache.

Three important strategies have been adopted to mitigate this difficult integration challenge. Firstly and most obviously, the CODAC group has developed an extensive set of standards that are to be used by plant system developers. They include specific hardware, software, naming convention, development tools, acceptance tests, documentation and procedures. These standards are bound together in the "Plant Control Design Handbook" (PCDH) which is made available to all plant system developers and potential vendors. It is an ITER mandate that they be followed, with enforcement occurring at design reviews.

Secondly, and notwithstanding that it represented an activity clearly not defined within its scope, the CODAC team created an I&C Support activity and dedicated considerable resources to it. They assigned liaisons and created a mechanism to work with the plant teams in the definition of interfaces and functional requirements. They have solicited and collected rack requirements for plant I&Cs. Recently this activity has been recognized in the baseline for the CODAC scope of work.

Thirdly, and from the outset, the proposed architecture envisaged a common interface between plant system I&Cs and CODAC [1]. This was to be effected by the use of a CODAC-supplied "Plant System Host" (PSH) which would front-end each plant system I&C using a mandated common protocol and software interface. The PSH is still a keystone element of the CODAC design, and EPICS channel access is the mandated common protocol [2]. CODAC will also provide developers with a "mini" version of CODAC for testing and development. The adoption of EPICS and Channel Access as the standard supervisory control infrastructure effectively assures a standard interface. The pieces appear to be in place for a seamless integration and commissioning phase.

WHAT COULD GO WRONG?

But here's the thing. There is very little incentive for Domestic Agencies and plant system developers to follow the standards. Rather, because systems are delivered "in kind," there is strong motivation for suppliers to do things as economically as possible; and it is almost always possible to build a specific system more economically – even better – using an optimized selection of hardware and software. Already there have been examples of DAs questioning the standards, or even stating that they do not intend to, or cannot, follow them. "Why do we have to use this over-specified PLC?" "…but we can use an existing design from our last project and save the design costs." "… but we need to spend our money domestically; your standard is only available overseas." "This plant system comes with its own control system – it can't be changed." Hard to argue with any of that.

Even if all the standards were to be followed, it is clear that there would be integration issues. Plant system acceptance tests will not have been done in the same environment as the final configuration. Network issues and conflicts can be anticipated. Use of rack space, already insufficient, can only be optimized centrally. Cable runs and grounding implementations will have to be adapted to the physical situation at Cadarache.

Because of the way procurements have been distributed, some systems have many more distinct I&C systems than is necessary or desirable. It is not clear who will integrate these systems, or where. This is a result of architectural design by "procurement arrangement" rather than by technical considerations. Optimization requires a centralized view.

The current implementation model is referred to as the "black box" model (you have no idea what's inside what you get) or sometimes the "thrown over the wall" model (you might catch it but you're not sure what to do with it.) There will be no trained on-site expertise to commission and maintain these systems. It seems that seamless integration is a pipe dream. Something similar, however, has been managed before.

THE SNS EXPERIENCE

The SNS construction project was a national collaboration of US Department of Energy Laboratories. Each delivered one or more major subsystems (Front End, Linac, Superconducting cavities, Storage Ring, Neutron Target) of the accelerator facility "in kind," and each of these subsystems was delivered with a control system that plugged into the common or "global" services. Like ITER, the Controls Group anticipated integration issues and responded with standardization and the selection of EPICS to be used throughout. But also like ITER, the project Work Breakdown Structure (WBS) placed control of the "plant system I&Cs" (to use ITER language) into the hands of the different partner laboratories, who had many good reasons for going their own way. There was no effective mechanism for enforcing standards, and no effective mechanism for optimizing on-site integration. This was the scenario presented to the SNS Conceptual Design Review (CDR) committee in 1997.

The Committee: "This isn't going to work."
SNS: "But we have standards and we have EPICS."
The Committee: "Good luck with that. Change it."

The SNS WBS was subsequently changed to place the control system at the top level and move both technical and financial responsibility for plant system I&Cs to the central group at Oak Ridge. In practice very little changed – the work was still distributed. Partner lab I&C teams remained at home and worked with local subsystem developers. Only now the funding passed through the central controls office in Oak Ridge, giving it the "clout" to enforce standards. Software was deposited weekly in a central repository at Oak Ridge, and built in that environment. Team leaders at the partner laboratories met frequently in the early days, and reported regularly later. Partner lab teams participated in the integration and commissioning in Oak Ridge, and some funds were set aside for their later intervention if required. Final integration went reasonably well.

APPLYING THE SNS MODEL TO ITER

In the fall of 2010, a proposal was made to ITER management to adopt an approach similar to that which had been successful at SNS. The interface to CODAC would move from a simple Ethernet connection to the PSH to the front panel of the I/O modules that are connected directly to the actuators and sensors of the plant systems. (These would remain the responsibility of the Domestic Agency and their plant system designers.) I&C teams would be formed at each of the Domestic Agencies to oversee and/or implement the plant system I&Cs. These teams would be funded as part of the centralized CODAC activities, and the team leaders would report to the CODAC Division leader at Cadarache. Equipment purchases for plant system I&Cs would be approved and funded centrally. Members of the DA I&C teams would participate in the installation, testing and eventual commissioning of their systems at Cadarache. The CODAC I&C Support team at Cadarache would consider issues related to optimization and integration of subsystems coming from more than one source and facilitate that task.

The cost of the plant system I&Cs (equipment, software and manpower) might be expected to be 50% - 70% of the total cost of the ITER control systems. (It was 60% for SNS.) As that work is not in the current budget for CODAC, a transfer of funds would be needed to finance this increase in scope. The mechanism proposed was a "tax" on each DA requiring I&C, calculated on the basis of anticipated costs of their plant I&Cs according to various "rules-of-thumb."

Although this proposal was put forward primarily as a risk mitigation, citing the risk of both cost and schedule overruns at integration time, it did suggest a potential cost saving to the DAs because of efficiencies that might accrue through bulk purchases and reduced hardware and space requirements. It was unfortunately this rather speculative cost saving, which would not in any case benefit the ITER Organization at Cadarache, that attracted the most attention, particularly as the proposal arrived just as a frenzy of cost containment was initiated at ITER.

The proposal generated considerable interest and discussion both at ITER IO and among the Domestic Agencies. Modified versions were advanced in which more hardware and software would be provided to plant I&C developers, but with no transfer of responsibility or funds. In the end the proposal was rejected for a number of reasons, perhaps most significantly because the "tax" was unacceptable. Worse, some stakeholders seemed to be of the opinion that plant I&Cs were already within the scope of CODAC in any case. That misinformed opinion forebodes an even greater train wreck.

THE PROPOSAL EVOLVES

As noted above, each DA is itself responsible for the design, construction, testing and delivery to Cadarache of a significant number of plant I&C systems. Europe, for example, will deliver the following systems, many extremely complex in themselves, most made up of multiple plant systems and including many plant I&C systems:

Two Cryogenic Plants (LN2 and 80K)
4 Remote Handling Systems
Tritium Plant Systems
Building Management Systems
Electrical Distribution Systems
Waste processing System
Test Blanket
14 Plasma Diagnostic Systems
3 Additional Heating Systems
Standalone Instrumentation
More…

This collection alone is probably comparable in range, cost and complexity to the complete SNS control system. One "additional heating system," for example, is a complete accelerator in itself, with ion source, RF systems, vacuum and cooling systems, beam line components, etc. It becomes immediately apparent that the integration issue presents to the DAs an (only slightly) smaller version of the same concerns that are perceived by CODAC.

For that reason, the controls team at the European DA last spring made a proposal similar to CODAC's for oversight, management and in some cases design and development of the plant I&C systems for which they are responsible. (Figure 1.) Like the CODAC proposal, they have proposed a small team of control system and project management experts to be located at the European DA offices in Barcelona. This team would take responsibility for oversight of all the plant system I&Cs that fall under the European purview. No black boxes. No walls. Like the CODAC proposal, they have appealed to the responsible DA ("Fusion for Energy," or "F4E") for funding to support this activity.

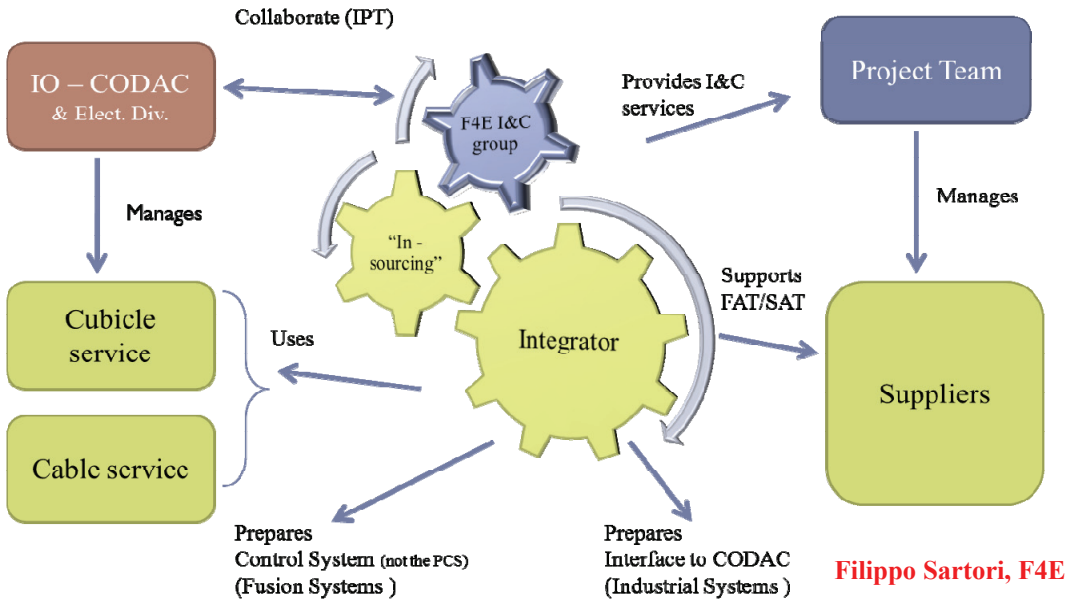


Figure 1. The European Proposal for plant I&C system development.

There are however important differences between the European proposal and the CODAC proposal. In this scenario, the DA controls team would be managed and funded from within the Agency and not through CODAC. Moreover, rather than using a significant number of F4E staff members, the European proposal suggests using a very small staff team and contracting most of the work to two external contractors: one "insourced" to supplement the staff in Barcelona and a second "outsourced" to an integrator that would perform much of its work wherever the subsystems are being built. (The CODAC proposal suggested using ITER staff members.) Formal approval is still required before this proposal becomes the official European approach to delivering plant system controls, but the idea of assigning most work to external contractors is consistent with the general ITER approach, and the proposal seems likelier to receive support for that reason. A meeting of potential integrators has already been held. Because this proposal advocates adherence to CODAC standards and recognizes the importance of a close collaboration with IO, the CODAC team is strong in its support.

CONCLUSIONS

Development of the ITER Control System is taking place in the presence of numerous constraints imposed by the imperatives of an international collaboration. Integration of plant control systems from many different suppliers distributed over the world presents a particularly difficult challenge. Centralizing the development of plant I&C systems was not accepted, however a compromise of centralizing development at the responsible DAs has been proposed by Europe. Learning to work effectively in this environment will be invaluable for future large international collaborations in science and other domains.

REFERENCES

[1] ITER CODAC Conceptual Design Report, ITER internal report
[2] Wallander et al, "News from ITER Controls – a Status Report", this conference

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

THE SESAME PROJECT

A. Nadji*, S. Abu Ghannam, H. R. Hoorani, Z. Qazi, I. Saleh, SESAME, Amman, Jordan
J.-F. Gournay, CEA/IRFU, Gif-sur-Yvette, France
E. Matias, G. Wright, CLS, Saskatoon, Saskatchewan, Canada
M. Heron, Diamond, Oxfordshire, U.K.
P. Betinelli-Deck, L. S. Nadolski, SOLEIL, Gif-sur-Yvette, France
B. Kalantari, PSI, Villigen, Switzerland

SESAME, P.O. Box 7, Amman 19252, Jordan

Abstract

SESAME (Synchrotron-light for Experimental Science and Applications in the Middle East) is a third-generation synchrotron light source under construction near Amman (Jordan), which is expected to become operational in 2015. SESAME will foster scientific and technological excellence in the Middle East and the Mediterranean region, build scientific bridges between neighbouring countries, and foster mutual understanding through international co-operation. The current members of SESAME are Bahrain, Cyprus, Egypt, Iran, Israel, Jordan, Pakistan, the Palestinian Authority and Turkey. An overview about the progress of the facility and the general plan will be given in this paper. We will then focus on the control system by explaining how this aspect is managed: the technical choices, the principal deadlines, the local staff, the international virtual control team, and the first results.

INTRODUCTION

Developed under the auspices of UNESCO and modelled on CERN, SESAME (Synchrotron-light for Experimental Science & Applications in the Middle East) is an international research centre under construction in Jordan [1].

It will enable world-class research by scientists from across the Middle East/Mediterranean region (in subjects ranging from biology and medical sciences through materials and environmental science and physics to archaeology), preventing or reversing the brain drain. It will also build bridges between diverse societies, contributing to a culture of peace through international cooperation in science. It will help foster closer links between people with different traditions, political systems and beliefs. The centrepiece of SESAME is a new 2.5 GeV third-generation light source which will provide very intense light from infra-red to hard X-rays for a wide range of studies. The microtron and the booster constituting the injector section were originally used at the former BESSY I synchrotron but have been greatly upgraded and refurbished. The energy of the microtron is 22.5 MeV whilst the electrons will be accelerated to 800 MeV in the booster. The storage ring will have an emittance of 26 nm.rad and 12 straight sections are available for insertion devices. The phase I scientific program has been finalized and foresees 7 beamlines providing wavelengths from infra-red to hard X-rays. The project is governed by a SESAME Council currently having nine members (Bahrain, Cyprus, Egypt, Iran, Israel, Jordan, Pakistan, the Palestinian Authority, and Turkey), whilst France, Germany, Greece, Italy, Japan, Kuwait, Portugal, Russia, Sweden, Switzerland, the UK, and the USA are observers. The project is benefitting enormously from help and advice provided by various synchrotron laboratories around the world (SOLEIL in France, ALBA in Spain, ELETTRA in Italy, the Swiss Light Source, Diamond in the UK and the Canadian Light Source). The capital funding needed to complete the construction of SESAME is being sought from the members and from various external sources.

Recent progress has included a growth in the number of technical and scientific staff to about 25 people [1]. The radiation shielding wall was completed in April 2011. Extraction of the first beam from the microtron has now been achieved at low energy (10 MeV) [2].

The current progress concerns tests of the booster equipment; the booster will shortly be installed in the tunnel, and preparation is under way for commissioning the booster with beam. In addition all the storage ring equipment has been designed, and the call for tenders is waiting for funding [2].

Figure 1: SESAME radiation shielding wall.

A SPECIAL PROJECT

SESAME is a wonderful project for the idea it conveys, but is very challenging because the budget is not secured in its entirety and arrives in small portions. The technical team is young and inexperienced in accelerator design and construction. The local industry also does not have direct experience in supporting such a project. As a result, the technical management of such a project cannot be conducted in a conventional way. The project leader must have a lot of imagination in order to overcome these challenges.

Thanks mainly to the International Atomic Energy Agency (IAEA) but also to other institutions [1], a programme of training has been underway since 2000. Most members of the SESAME team have spent several months in one of the world's synchrotron radiation facilities and learned about several areas of synchrotron light source construction and operation. Experts from these laboratories have also visited SESAME and this has had a positive impact on the local team. In the case of the control system, a virtual international group has been created.

This has all helped to achieve some important milestones such as the successful commissioning with beam at low energy of the microtron, testing of all of the booster subsystems (RF, vacuum, and cooling systems, diagnostics, and so on), and providing new booster power supplies and new booster dipole vacuum chambers after an international call for tenders. Good news is foreseen concerning the budget. Commitments and offers announced and confirmed by Members in May 2011 look set to provide most of the remaining $35 million needed for the storage ring and the three beamlines that will be in operation from day 1.

TEAM BUILDING

The International Control System Virtual Team

The recruitment of the control system group and its management was particularly difficult. The potential candidates are young and unfamiliar with accelerator control. There is also a limited budget which prevents the hiring of a significant number of staff or the outsourcing of work to specialized companies. SESAME has found very good candidates who have electronics or software skills, but who are new to accelerator or light source control systems. Each newly-hired engineer has therefore been trained in one of the following laboratories: SOLEIL, SLS, CLS, and/or DLS; and coached in order to be operational as soon as possible. Although SESAME, in common with other light sources, has advisory committees which meet once or twice a year, a need for senior experts who can follow the work more closely was essential.

At the end of 2010, the SESAME project leader decided to set up an international "virtual" control system group. This group is composed of experts and managers in the field of accelerator control from the Canadian Light Source (CLS), Diamond Light Source (DLS), the Institute of Research into the Fundamental Laws of the Universe (IRFU) of the CEA, Synchrotron SOLEIL, the Swiss Light Source (SLS), and of course from SESAME itself. The idea is to have close coaching of the SESAME control system team.

A Cross-Over Management

Inside the virtual group management task and responsibilities are defined for better efficiency. The coordination of the virtual group is handled by Pascale Betinelli from SOLEIL. The main reason for this choice is to make the contact between SESAME and the virtual group easier; the part time Technical Director of SESAME (until the end of this year) is the current Director of SOLEIL Sources and Accelerators Division. Moreover Mark Heron, the DLS member of the virtual control group, is present on the Technical Advisory Committee (TAC) of SESAME and on the Computing Advisory Committee (CAC) of SOLEIL. Whilst each local engineer is in charge of a part of the control system, each expert member is responsible for following that part closely and teaching the local engineer particular technical aspects of control.

Figure 2: Members of the SESAME International "Virtual" Control System group.

Recruitment

The coaching starts with recruitment: the members of the virtual group guide SESAME's leaders by finding the right technical skills in the CVs. SESAME's leaders then conduct each job interview and select the person who fits with the technical requirements and the motivation of the project. Today the local control system team is composed of three engineers: Saed Abu Ghannam from Palestine, Ibrahim Saleh from Jordan, Zia-ul-Haque Qazi from Pakistan.

Training

Each engineer had good electronics or software skills but needed to be trained on accelerator control systems. First they went around the world to learn the basics of instrumentation and control: EPICS, VME and Timing System. Saed Abu Ghannam spent one month at CLS, attended the PCaPAC 2010 workshop then spent two months at SLS, one week at SOLEIL and visited IRFU. He was trained in EPICS and discovered the different aspects of a synchrotron radiation facility. Similarly, Ibrahim Saleh has been invited to ICALEPCS 2011 as a member from an industrially emerging nation and will take this opportunity to spend a few weeks at DLS to learn EPICS and interlocks, and at SOLEIL, where he will stay two weeks working on PLC aspects and the installation process. The third member of the SESAME control system group, Zia-ul-Haque Qazi, should soon undertake similar training.

Coaching

Day by day the virtual group guides the control team in the development process. They explain each step to them, providing examples and a work plan. To help the local team, the virtual group has provided a list so that they can easily find who to contact for each type of problem or for advice. On a regular basis the virtual group organizes video conference meetings with all the members, in order to validate technical choices or to review sub-projects such as power supply control, vacuum control, etc. The local engineer who is in charge of a particular task prepare the work with the person responsible for SESAME subsystems and then discuss it with the virtual group in a dedicated video conference meeting. The requirements and the control architecture are then discussed during the review.

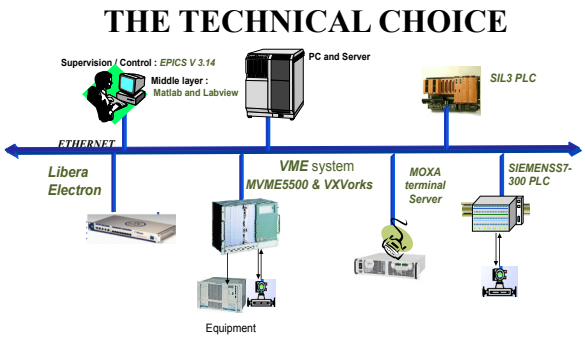


Figure 3: Control Architecture.

SESAME needs to have the most up-to-date control possible, but in doing so needs to consider how the solutions will be supported. It is therefore very important to select items corresponding to the skills of the virtual group. Sometimes SESAME has the chance to receive gifts from the international community. These generous gifts help us to reduce costs but involve some difficult upgrades due to the age of the equipment given or to technology changes. The analysis of the feasibility of integrating old technologies is a process that requires time and special skills

All the time we have to make a compromise between up-to-date technology and pragmatism.

As a result of following these guidelines, the main technical choices are very conventional and tested: EPICS and VME have become established. Figure 3 shows a schematic view of the main architecture.

VME System

We have selected EPICS 3.14 in order to benefit from the latest improvements. For VME systems, we first analyzed the feasibility of integrating old VME 2304 CPUs. Finally we selected the boards currently used at SLS, DLS, and IRFU, namely MVME5500 from Emerson and I/O boards from Hytec Electronics. We would have liked to use the very latest EMERSON CPU technology but during the startup period we cannot accommodate the work load of adopting this new technology. On the VxWorks kernel side, we have chosen to implement version 6.9. This allows us to take advantage of the know-how of IRFU, which is in charge of building the appropriate control system environment (VxWorks kernel with required functionalities, EPICS3.14 base, extensions and development framework).

Because VME technology is expensive and complex to implement and maintain, the international experts encourage the use of stand-alone PLC and Ethernet modules as far as possible.

PLC Systems

SIEMENS PLCs are the de facto standard as SESAME has experience in this field. The local team has already upgraded the control of the Microtron with a CPU312. Furthermore, SOLEIL and DLS have experience with this type of PLC system.

In most cases the machine protection systems will be connected to the PLC from different subsystems (vacuum, cooling, etc.). Part of the power supply and vacuum control will be performed by PLC. In accordance with the SOLEIL specification method, an order has already been placed with SIEMENS' Jordanian supplier. In October SESAME will start to implement booster development with the help of SOLEIL.

Ethernet Standalone Equipment

Serial links are communication interfaces that will have to be managed in large numbers. To reduce cost we have selected a MOXA terminal server with the first items ordered from a local supplier in Dubai. The terminal server was received and an implementation for the corrector power supply of TL1 has already been done. Thanks to the training already received, the implementation required very little help. Figure 4 shows the first EDM interface for the corrector power supply.

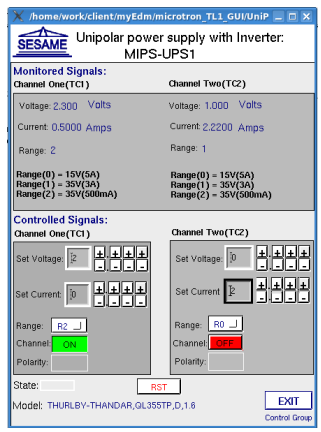


Figure 4: EDM interface for corrector power supply.

The control of diagnostics is supervised by CLS, which has provided a template and advice. Many diagnostics devices come from BESSY I. The three fluorescent screens available around the Booster have been refurbished and the cameras have been replaced by a low-cost Ethernet camera. Current measurement performed by a cavity is still under discussion to understand the

appropriate control mechanism. Whilst there was no tune measurement in the old BESSY I Booster, it is planned to have such a system at SESAME. The BPM read-out electronics is performed by I-Tech Libera Electron modules. SESAME has already received seven Libera Electron modules to be used in its booster ring. A SESAME engineer is in charge of the implementation of the embedded process. Presently EPICS IOC (from DLS) has been installed in one of these modules and the module and driver performance is being analyzed. Performance analysis is being carried out with a 500 MHz signal provided to the Libera Electron inputs through a clock splitter. However the available EPICS IOC is incompatible with latest version of Libera Electron and there is a need for modifications which are currently under way. On the other hand, GUIs have also been obtained from DLS to receive and view information from Libera Electron modules, and these are being used for the performance analysis of the EPICS driver.

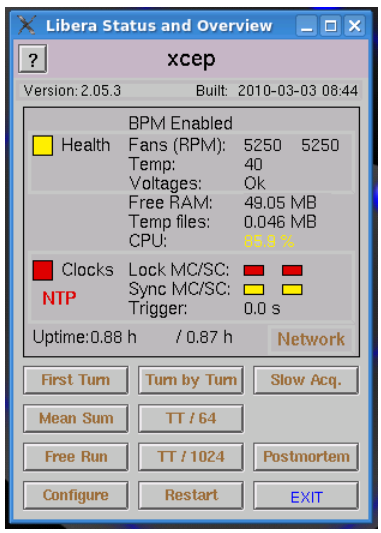


Figure 5: Main GUI for Libera Electron.

Figure 5 shows the main screen from the GUIs for Libera Electron. It shows the health status, the clock status, controls to obtain various types of data from Libera modules such as First Turn, Turn by Turn, etc., as well as controls for Configuration and Restart. Once the performance of EPICS IOC and GUIs is finalized, they will need to be customized by SESAME according to its naming convention.

Booster Timing System

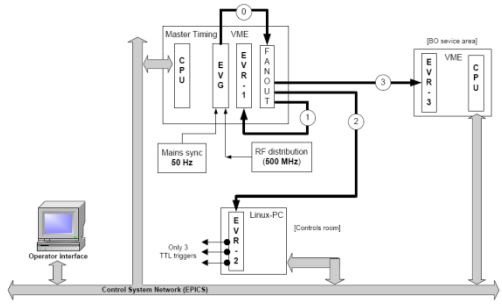


Figure 6: Booster timing system.

SESAME's booster timing system is based on the global event system, widely used in many light source facilities such as APS, SLS, DLS and others. The booster timing system has been designed and developed by SLS, and all the hardware used is from Micro-research Finland. In the SESAME booster timing system, one event generator and three event receivers will be used in order to provide and distribute all the event sequences to operate the booster. Figure 6 shows a schematic of the booster timing system. It will be delivered to the SESAME site at the end of November to be tested before its deployment.

End User Development Environment

The main development environments used are Labview and Matlab. These two software packages are easily interfaced to the EPICS framework and offer a suitable end-user environment. Matlab middle layer physics applications [3] will be used as commissioning tools to and for a few high level control applications from the transfer line to storage ring as in similar facilities. With Labview we can directly integrate the prototype of the Booster RF control, which has already been completed using a CPCI system. In a second step we will upgrade this prototype to run on a VME platform to provide consistency with the rest of the control system.

Network and Server

In SESAME, the control servers are based on virtual machines running Scientific Linux. We currently have around 15 powerful virtual machines. Those machines will be used in making EPICS servers, development consoles, file systems and others.

HOW TO SET UP THE RELATIONSHIP WITH MANUFACTURERS AND SUPPLIERS

Industry in the Middle East has very little experience of Synchrotron light sources. There is also a limited presence of the big companies involved in control systems, and those which are present are reluctant to get involved. From this low point, SESAME has to educate and develop relations with potential suppliers.

International support is again essential: Day by day, through discussion after discussion, companies will come

to understand the approach and objectives. Conversely, SESAME engineers must understand and adjust to the ways in which western companies and people think and work. Once mutual understanding has been established, companies are often willing to help and support the SESAME project. For instance Windriver has offered to let SESAME join their research program.

HOW TO SET UP THE INSTALLATION PROCESS

SESAME engineers have no experience of the requirements and the complexity that they will face during installation. Like all people in similar positions, they need time to understand the necessity for and the strategic requirements of aspects such as a naming convention and a cabling database, and the need to do some tasks in advance. To tell them is not always enough: they need to acquire experience in order to appreciate every detail of the virtual team's advice. Many short discussions are necessary to compensate for the drawbacks of having a virtual team: E-mail and documents are not enough; direct contact is very important to solve problems.

Today, reviews have already been conducted for the power supply, vacuum and diagnostics areas.

OUTCOME AND FUTURE

We should always keep in mind that SESAME is not a standard project and the context and the issues are complex. In 2009 the TAC was worried about the state of the control system. This year, after just one short year of existence of the control system virtual group and the recruitment of local engineers, control system development is going ahead at SESAME. We hope to assemble most of the booster control system by the end of the year. Some control parts like the pulsed magnet control are still critical because of the age of the equipment but we are confident that we can solve the problems.

Unfortunately we note that planning has to be flexible to handle this complex project. The booster assembly will be a true test to demonstrate to the international community the legality and viability of SESAME.

ACKNOWLEDGMENTS

We wish to thank:

- Elder Matias and his team from CLS
- Mark Heron and his team from DLS
- Jean-François Gournay from IFRU
- Babak Kalantari from SLS
- Pascale Betinelli and her team and Laurent S. Nadolski from SOLEIL
- SESAME subsystems responsible for valuable discussions.

They are very involved in this project and always give us good advice and training.

REFERENCES

[1] www.sesame.org.jo.
[2] A. Nadji & al., IPAC'11 proceedings
[3] J. Corbett, G. Portmann, and A. Terebilo, ``Accelerator control middle layer'', PAC'03, May 2003, Portland, pp.~2369--2371.

THIRTY METER TELESCOPE ADAPTIVE OPTICS COMPUTING CHALLENGES

C. Boyer, B. Ellerbroek, L. Gilles, L. Wang, TMT Observatory, Pasadena, CA, USA
G. Herriot, J.-P. Veran, HIA, Victoria, BC, Canada
G. Hovey, DRAO, Penticton, BC, Canada
S. Browne, tOSC, Anaheim, CA, USA

Abstract

The Thirty Meter Telescope (TMT) will be used with Adaptive Optics (AO) systems to allow near diffraction-limited performance in the near-infrared and achieve the main TMT science goals. Adaptive optics systems reduce the effect of the atmospheric distortions by dynamically measuring the distortions with wavefront sensors, performing wavefront reconstruction with a real time controller (RTC), and then compensating for the distortions with deformable mirrors. The requirements for the RTC subsystem of the TMT first light AO system will represent a significant advance over the current generation of astronomical AO control systems. Memory and processing requirements would be at least 2 orders of magnitude greater than the currently most powerful AO systems using conventional approaches, so that innovative wavefront reconstruction algorithms and new hardware approaches will be required. In this paper, we will first present the requirements and challenges for the RTC of the first light AO system, together with the algorithms that have been developed to reduce the memory and processing requirements, and then two possible hardware architectures based on Field Programmable Gate Array (FPGA).

INTRODUCTION

The Thirty Meter Telescope (TMT) Project [1] is designing and building a thirty-meter diameter telescope for research in astronomy at optical and infrared wavelengths. The core of the TMT is a wide field, altitude-azimuth Ritchey-Chretien telescope with a primary mirror consisting of 492 segments. Instruments are located on two large Nasmyth platforms, addressed by an articulated tertiary mirror.

The initial Adaptive Optics (AO) architecture for the TMT is defined to provide near-diffraction-limited wavefront quality and high sky-coverage in the near infra-red (IR) for the first light TMT science instruments IRIS, a near-infrared instrument with parallel imaging and integral-field-spectroscopy support; and IRMS, an imaging, multi-slit near-infrared instrument. The initial AO architecture is a Laser Guide Star (LGS) Multi Conjugate AO (MCAO) architecture consisting of (i) the Narrow Field IR AO System (NFIRAOS) [2], which senses and corrects for wavefront aberrations introduced by the atmospheric turbulence and the telescope itself, and (ii) the Laser Guide Star Facility (LGSF), which generates a constellation of LGS in the mesospheric sodium layer with the brightness, beam quality and geometry required by both NFIRAOS and the future second generation of TMT AO systems [3].

The NFIRAOS system includes two 60x60-order Deformable Mirrors (DM) conjugated at 0km and 11.2km, one fast Tip-Tilt Stage (TTS) serving as a mount for the ground level DM, six 60x60-order LGS Wavefront Sensors (WFS), one high-order Natural Guide Star (NGS) WFS for non-LGS operations, up to three low-order NGS WFS working in the near–infrared and located within each NFIRAOS instrument (also referred as the On-Instrument WFS or OIWFS), and a Real Time Controller (RTC) processing the inputs from the multiple WFS to compute the commands to the deformable mirrors and the tip/tilt stage at sampling frequencies up to 800Hz.

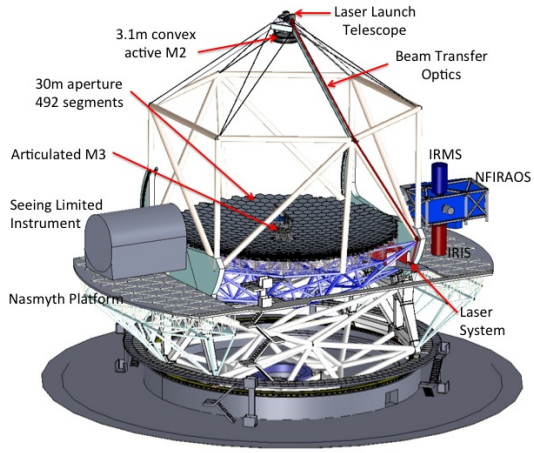


Figure 1: TMT telescope overview and first light instrumentation.

NFIRAOS RTC REQUIREMENTS AND CHALLENGES

The NFIRAOS RTC is one of the most challenging computing components of TMT [4]. It includes several modes of AO operation. The mode that has the most demanding requirements is the LGS AO operation mode. A block diagram of this mode is given in Figure 2.

The RTC requirements in the LGS AO operation mode are split into two categories:

- The *hard real time requirements*, which consist of the LGS wavefront sensor pixel processing, the LGS reference processing, the On-Instrument wavefront sensor pixel processing, the very high-order LGS tomographic wavefront reconstruction using measurements from these multiple wavefront sensors, the On-Instrument wavefront reconstruction

and the calculation of the two deformable mirrors and tip-tilt stage commands. These processes are operated at up to a 800Hz sampling rate, with a 1000µs latency (and a strong goal of 400µs).

- The *background and optimization requirements*, which operate at slower sampling rates to i) optimize in real time the parameters of the hard real time processes as the observing parameters and atmospheric conditions change, ii) estimate the turbulence parameters, iii) offload persistent, low spatial frequency components of the deformable mirrors and tip/tilt stage commands to the telescope, iv) compute the commands for the Fast Steering Mirror located within the Laser Guide Star Facility (LGSF) and, v) acquire the data necessary to reconstruct the AO-compensated science PSF in post-processing (compute AO-compensated science PSF as a goal).

Table 1: RTC Key Numbers

Item	Requirement
Number of LGS WFS	6
Number of pixels per WFS	204,792
Number of gradients per WFS	5792
LGS frame rate	800Hz
Full frame readout time per LGS WFS	500µs
LGS WFS pixel processing latency (performed synchronously with the digitization of the LGS WFS pixel intensities)	10µs
Number of DM actuators	7673
Latency from last gradient to last DM command	1000µs (goal of 400µs)
RTC telemetry storage	90TB (goal of 140TB)
RTC telemetry required data rate	3.5GB/s (goal of 5GB/s)
PSF statistical data required data rate	60MB/s
RTC maximum power dissipation	1500Watts
RTC telemetry storage maximum power dissipation	6000Watts

The RTC works in synchronization with the Reconstructor Parameter Generator (RPG), which sole tasks are to initialize all of the RTC hard real time parameters, and update in real time the wavefront reconstructor parameters and temporal filters based on the RTC inputs. The RPG is also in charge of monitoring the performance of the AO system during observations and providing the tools necessary to calibrate the AO system during day-time calibrations.

Some aspects of this architecture, which have received considerable attention over the last years include the implementation of:

- The LGS WFS and OIWFS "matched filter" gradient estimation algorithms within the LGS WFS and OIWFS pixel processing processes;
- The "split tomography" wavefront reconstruction algorithm, which decomposes the atmospheric turbulence profile into two orthogonal subspaces, which are estimated and controlled separately using the On-Instrument and LGS WFS measurements;
- The real-time estimation of the turbulence profile and atmospheric parameters using slope detection and ranging method (SLODAR);
- The temporal filters and telescope offloads in the deformable mirrors and tip/tilt stage control processes.

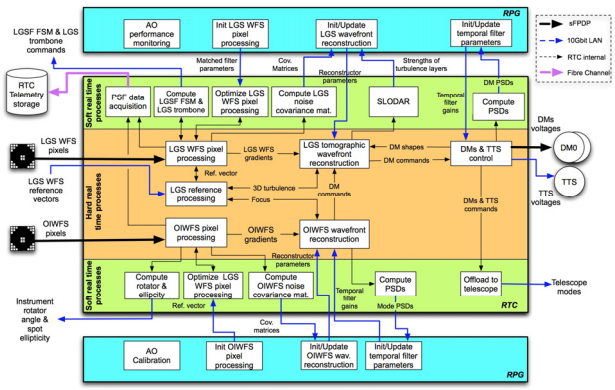


Figure 2: Top-level RTC control block diagram for the LGS AO operation mode. The RTC works in synchronization with the Reconstructor Parameter Generator (RPG – blue boxes). The RTC processes are split into two categories: the hard real time processes (orange) and the background and optimization processes (green). Finally, but not least, the RTC design should be modular to allow the system to be modified or upgraded for the next generation of AO systems.

Wavefront Pixel Processing

The LGS WFS pixels are processed using a constrained matched-filter algorithm. It is a noise optimal algorithm, which allows a reduction in the laser power requirements compared with a classical center-of-gravity algorithm. It consists of a simple matrix-vector multiplication performed synchronously with the digitization of the pixels intensities. The matched filter algorithm is updated in real time at a 1Hz sampling rate (goal of 10Hz) to account for changes in seeing, sodium layer profile and laser beam quality. The optimization is performed by dithering the Laser Guide Star Facility fast steering mirrors.

The On-Instrument WFS pixels are processed using a constrained matched filter algorithm as well. The matched filter algorithm is updated in real time at a 0.1Hz

sampling rate based upon variations in seeing and AO system performance.

Computationally Efficient Wavefront Reconstruction

The NFIRAOS wavefront reconstruction problem requires the computation of over ~7700 DM actuator commands from about ~35,000 LGS WFS measurements at a frame rate of 800 Hz. The standard matrix-vector-multiply (MVM) solution becomes very impractical for systems of this dimensionality, particularly if the control matrix must be updated in real time to account for changes in the atmospheric turbulence profile, rotation of the TMT pupil, or other time-varying effects, which would require the inversion of a very large matrix in real time. Computationally efficient algorithms must be implemented instead. Generally speaking, these algorithms implement close approximations to minimum variance atmospheric tomography. These tomographic algorithms are performed in two steps: estimation of the atmospheric turbulence profile from the LGS WFS measurements, and then projection onto DM locations (least-squares DM fitting). Five low-order modes are computed at lower bandwidth from the On-Instrument WFS wavefront reconstruction using a noise-weighted least-squares reconstruction control matrix. These modes are converted into DM commands and integrated with the DM commands computed from the LGS measurements (split tomography).

Four algorithms have been studied for the tomography step and they all meet the required AO performance in terms of wavefront errors: (i) 30 iterations of Conjugate Gradient without preconditioning (**CG30**), (ii) 3 iterations of Conjugate Gradient with a Fourier Domain Preconditioning Hermitian Matrix (**FD-PCG3**), (iii) Block Gauss-Seidel with 20 iterations of Conjugate Gradient for each layer (**BGS-CG20**) and (iv) Block Gauss-Seidel with Cholesky Back Substitutions for each layer (**BGS-CBS**). Note that closed loop convergence of all these algorithms is accelerated by using warm restart. Finally, the DM fitting step is performed using 5 iterations of Conjugate Gradient. Each proposed algorithm can be expressed as a combination of sparse matrix multiplication, geometrical wavefront propagation through square grids, Fourier transforms, and/or Cholesky back-substitution through triangular sparse matrices.

Finally, the LGS WFS reconstruction parameters are updated in real time at a 0.1Hz sampling rate.

DMs and TTS control

A simple integrator filter with an adjustable gain is applied to the DM error signals computed by the wavefront reconstruction processes. A woofer/tweeter algorithm is implemented for the control of the tip/tilt modes: the TTS commands are obtained by applying an additional proportional-integrator filter to the tip/tilt components of the filtered ground-layer DM commands. The filtered DM and TTS commands are clipped to avoid saturation, and integrator windup is prevented by subtracting such clipping adjustments from the inputs of the temporal filters.

RTC Memory and Computation Requirements

The memory and computation requirements for the LGS WFS pixel processing and the LGS wavefront reconstruction processes are presented in Table 2. These are the most demanding RTC requirements. The LGS wavefront reconstruction memory and computation requirements are presented for the four algorithms described above. Two byte fixed-point arithmetic has been used to estimate the memory requirement for the LGS WFS pixel processing requirements and four byte floating-point arithmetic has been used to estimate the memory for the LGS wavefront reconstruction process. These requirements are useful, but not sufficient to demonstrate that a specific parallel hardware architecture meets the TMT requirements. Data transfer required between the processing elements should be carefully analyzed and minimized when designing the hardware architecture to avoid stall issues.

Table 2: RTC Computation and Memory Requirement

	Memory (MB)	**Nb. of Op. GMAC/s (1000μs latency)**
LGS WFS processing	10	7
LGS wavefront reconstruction		
BGC-CBS	50	80
BGS-CG20	2	280
CG30	2	245
FD3 (2 layers oversampled)	10	140

RTC CONCEPTUAL DESIGN STUDIES

Two RTC conceptual design studies were conducted for TMT in 2009. One study was lead by Dominion Radio Astrophysics Observatory (DRAO) and also included HIA, Lyrtech and the University of Victoria [5]. The second study was performed by the Optical Sciences Company (tOSC) with support from Montana State University. Both groups developed successful designs meeting all performance requirements, and in some cases many goals, for the NFIRAOS RTC. Both studies implemented the processing algorithms specified by TMT in designs based upon existing field programmable gate arrays (FPGAs) and digital signal processors (DSPs), and in electronics packages meeting the requirements for rack space, mass, and power dissipation. The proposed hardware architectures have similarities, but depend greatly upon the choice of tomographic algorithm, which impacts the processing and memory requirements.

DRAO Conceptual Design

A block diagram of the conceptual design proposed by DRAO is given in Figure 3.

The hardware architecture consists of nine custom FPGA boards each including six Xilinx Virtex-5 FPGA, two custom interface boards with thirty-two sFPDP full duplex-links for communication with AO components and RTC telemetry storage system (referred as data recorder in Figure 3) and two general purpose computer boards. The boards are mounted within an Advanced Telecommunications Computing Architecture (ATCA) chassis. The system is highly modular and meets the TMT latency requirement using fixed-point arithmetic and the Block Gauss-Seidel with Cholesky Back Substitutions algorithm. The high-speed WFS pixels are received by the interface boards and then distributed to two FPGA boards, which compute the WFS gradients. The WFS gradients are then forwarded to the wavefront reconstruction engine, which consists of seven FPGA boards and which computes the DM commands. The DM commands are then forwarded to the interface boards, and then applied to the DMs. Scaled down versions of the processes were implemented on Xilinx FPGA to demonstrate the processing time and validate the fixed-point operations.

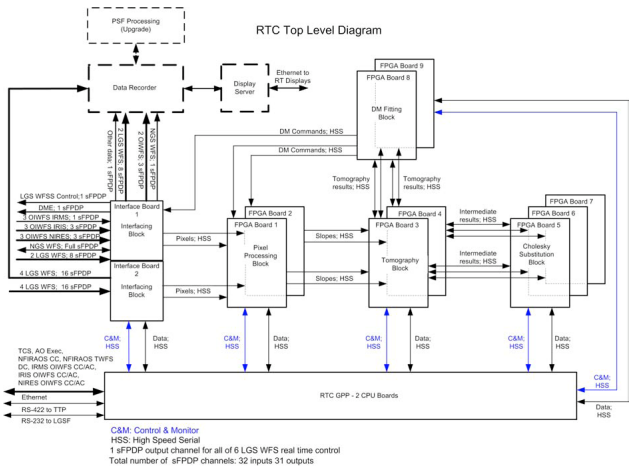


Figure 3: DRAO proposed conceptual design.

tOSC Conceptual Design

A block diagram of the conceptual design proposed by tOSC is given in Figure 4.

The hardware architecture consists of seven TigerSHARC cluster boards, each equipped with eight TigerSHARC DSPs and one Xilinx Virtex-5 FPGA, four FPGA cluster boards each equipped with four Xilinx Virtex-5 FPGA and one TigerSHARC DSP, and one general purpose CPU board. The boards are mounted within an ATCA chassis. The proposed architecture meets the TMT latency goal requirements using floating-point operations and the Conjugate Gradient without preconditioning algorithm. The TigerSHARC boards are used to handle the WFS pixel processing (calibrations, gradient computations and matched filter updates, etc…). The FPGA boards handle the LGS wavefront reconstruction and associated background and optimization tasks. Key functions of the selected algorithm and data transfers were tested on Xilinx FPGA evaluation boards.

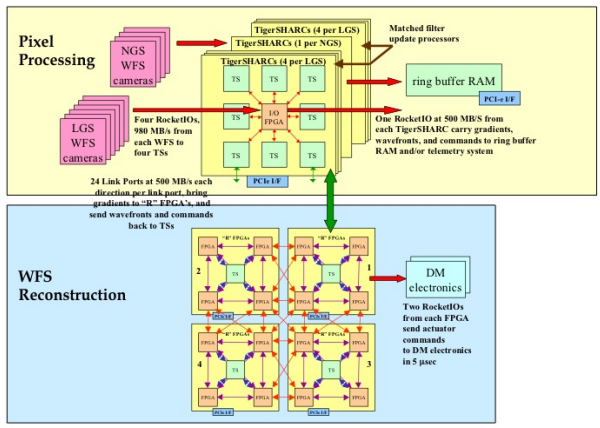


Figure 4: tOSC proposed conceptual design.

CONCLUSION

The TMT NFIRAOS RTC requirements are challenging not only because of the computing and memory requirements, but also because of the complexity of the algorithms to implement, and the number of interfaces to manage. We have demonstrated that the RTC can be implemented via modular and highly parallel hardware architectures, which use existing computing technologies. The next steps for the RTC will be to review the latest generation of multi-processors (Xilinx Virtex-6, Nvidia GPU…), then to define the hardware architecture for a selected algorithm, and finally to prototype and test key components of the RTC hardware architecture.

REFERENCES

[1] G. Sanders and J. Nelson, "The status of the Thirty Meter Telescope project", Proceedings of SPIE 7733-69 (2010).

[2] G. Herriot, D. Anderson, J. Atwood, C. Boyer, P. Byrnes, R. Conan, B. Ellerbroek, J.T. Fitzsimmons, L. Gilles, P. Hickson, A. Hill, K.J. Jackson, O. Lardiere, T. Pfrommer, J.P. Veran, L. Wang and Yvan Wevers, "NFIRAOS facility adaptive optics system for the TMT", Proceedings of SPIE 7736-9 (2010)

[3] L. Simard, C. Boyer, D. Crampton, and B. Ellerbroek, "The TMT Instrumentation Program", Proceedings of SPIE 7735-73 (2010).

[4] C. Boyer, L. Gilles, B. Ellerbroek, G. Herriot, J.P. Veran, "Update on the TMT adaptive optics real time controller", Proceedings of SPIE 7015 (2008).

[5] G.J. Hovey, R. Conan, F. Gamache, G. Herriot, Z. Ljusic, D. Quinn, M. Smith, J.P. Veran and H. Zhang, "An FPGA Based Computing Platform for Adaptive Optics Control", Proceedings of first AO4ELT Conference http://ao4elt.edpsciences.org, (Paris, 2009).

IMPROVING DATA RETRIEVAL RATES USING REMOTE DATA SERVERS*

Ted D'Ottavio, Bartosz Frak, Seth Nemesure and John Morris, Brookhaven National Laboratory, Upton, NY, U.S.A.

Abstract

The power and scope of modern Control Systems has led to an increased amount of data being collected and stored, including data collected at high (kHz) frequencies. One consequence is that users now routinely make data requests that can cause gigabytes of data to be read and displayed. Given that a users' patience can be measured in seconds, this can be quite a technical challenge. This paper explores one possible solution to this problem - the creation of remote data servers whose performance is optimized to handle context-sensitive data requests. Methods for increasing data delivery performance include the use of high speed network connections between the stored data and the data servers, smart caching of frequently used data, and the culling of data delivered as determined by the context of the data request. This paper describes decisions made when constructing these servers and compares data retrieval performance by clients that use or do not use an intermediate data server.

INTRODUCTION

Can a system be constructed that can read and display tens or hundreds of millions of stored data points to a user in a reasonable period of time (< 10 seconds)? What are the limitations and how can they be addressed? These are the questions that led to the research being reported on here.

The Controls logging system within the Collider-Accelerator Department at Brookhaven National Laboratory collects and stores measurement and setting data for offline analysis. The logging system, in place for about 10 years, uses a combination of user-generated and system-generated requests. These requests are passed to dedicated logger processes, which store the data to disk along with database records indicating how to map the requests to the stored data files. Data is retrieved and displayed by specialized software that allows a user to see what data was logged and to select data to be displayed.

The BNL logging system has become increasingly popular over the years, with the amount of data collected growing by a factor of 10 in the last 5 years, and expected to grow by another factor of 10 over the next 3 years (12 TB of time-series data was stored last year). Much of this increase is coming from requests to store data collected at high frequencies (720Hz to 10 kHz). In rough terms, about 500 MB of data is stored per day for every 1 kHz parameter requested. Retrieving and displaying a days worth of data for just a couple of these parameters means processing about a GB of data contained within files many times that size. For thick client applications that get data directly from the file system, this means transferring all of the data over the network to the client applications, picking out the requested data, and displaying it to the user before his or her patience has been exhausted. This puts a severe strain on both the network (to read the data) and the plotting software (to display it).

The solution reported in this paper uses a data server - a combination hardware/software solution. Its job is to handle user requests for logged data in the most efficient way possible. Its efficiency is a result of two key ideas:

- Read the data quickly – improve network speed to stored data, parallelize reads, and cache data.
- Cull the returned data - send the user only the data that can reasonably be distinguished on the display.

The use of an intermediate data server here follows similar efforts in place within the EPICS community [1] and at the CERN/LHC [2].

SYSTEM DESCRIPTION

The term data server denotes a collection of enterprise grade middleware applications deployed in a virtual cluster of dedicated and adopted process servers. At the core of the system is a single purpose application, whose role is to handle client requests for stored data. These requests are divided into discrete tasks, which are processed in parallel either locally or, in the event the core server cannot complete assigned workload, on one or more satellite servers. Result fragments from all tasks belonging to the same request, are reassembled by the core server and delivered back to the client.

Software Architecture Overview

Java Enterprise Edition 6 (Java EE6) was chosen as a base platform for the entire system. Benefits include a wide array of available web technologies [3] and a scalable business logic platform as well as built-in management features. Glassfish 3.1, which provides a complete open source Java EE6 reference implementation [4], was selected over other EE6 compliant applications servers.

The enterprise applications used to construct the data server are divided into three logical modules (Figure 1):

- Web component, which exposes a RESTful web service API to the remote clients.
- Request dispatch and assembly (RDA) Enterprise Java Beans (EJB) business module responsible for database communication, task scheduling and construction of the response objects.
- Request extract and transform (RET) EJB business module responsible for the data collection, caching and transformation. This component is divided

*Work supported by Brookhaven Science Associates, LLC under contract no. DE-AC02-98CH10886 with the U.S. Department of Energy

furthermore into local and remote sub-modules - the latter deployed on the satellite servers.

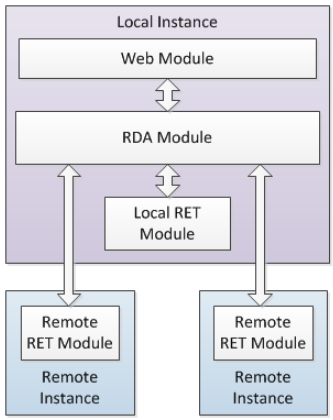


Figure 1: High-level logical system structure.

Request Lifecycle

Every request starts with a RESTful GET call to a designated web service method. The URL used to identify the requested resources has one mandatory path element as well as three mandatory and one optional parameter values – clients consuming this web service are required to specify a name of the request file, start and end times as well as at least one data item name contained in the requested file. The web module submits the request to the RDA component, which queries one or more databases for a list of file paths that fall between the specified start and end time values and match exactly the request file name. At this stage the RDA module attempts to locate each file / resource pairs in either the local or one of the remote caches. Cached file elements are immediately queued on the owning instance for final processing (i.e. culling). The remainder of the file list is sent to a scheduler on a local instance, which attempts to distribute the combined read and process tasks between the clustered RET modules by taking into consideration both performance as well as coherence aspects of the request. The factors, which influence scheduler's decision include:

- The queue length on each cluster instance – the shorter the wait time, the greater the chance that the scheduler will pick that instance.
- Instance spatial location – some cluster member may be physically closer to the stored data than others (i.e. application could be deployed directly on one of the archive server host), which allows them to read stored data at a much higher rate than their networked counterparts.
- Request length – shorter requests, which estimated processing time of under 5 seconds benefit from being scheduled on only one instance.

Regardless of which instance or instances were involved in reading and/or processing the scheduler's request, all intermediate results always make it back to the local instance for final assembly. During this phase of the request lifecycle the results from each processing task are checked for errors. Additionally, if they were processed out of order, they have to be rearranged according to the time index given to them by the scheduler. At this stage the RDA module returns the time sorted, processed results back to the Web component, which can package the processed data in a JAXB compliant wrapper and ship it back to the client.

Spotlight on Culling

Every request that passes through the data server is subjected to the culling algorithm. This algorithm was designed to cut down high volume, time domain datasets to more manageable, lower-density sets. The goal is to return a dataset to the user that is virtually indistinguishable on a scatter plot to the full, un-culled dataset. We have found for typical monitors and resolutions that this occurs when a dataset of about 20k points is returned. The advantage here is that the culled dataset can be transported to the client much more quickly than the full dataset. The disadvantage is that any change in the plot axis limits (for example, a zoom) forces a new data retrieval. Our experience has been that these zoom slowdowns are minimal (< 1 sec) as long as the original data is cached on the data server. The culling algorithm incorporated into the data server uses a parallel processing pipeline, allowing the server to process up to a billion points per second. An enhancement to this algorithm will incorporate user-selected data filters that can be used to further refine the returned datasets.

Hardware

The central EJB container runs in a 64bit environment (Red Hat EL6) on a dedicated rack mounted system. Two 6 core Xeon CPUs, each running at 3.47Ghz, and a total of 144GB of RAM is available to one or more virtual machines - this number depends on garbage collection issues we might encounter once the system is put into production. The SSD caching subsystem is made up of four Intel 250GB solid state drives configured in RAID0 on a 6Gb/s LSI controller. The SSD array is expandable to 3.5TB, and like RAM can be split into multiple partitions. Four 1Gb/s bonded adapters handle the network access to data stores, though only one was used for the results that follow. Satellite servers run in smaller virtual machines on equally powerful CPUs. They have their own in-memory cache, but they lack the disk-store caching, which is available on the core machine. Figure 2 shows the data server's communication diagram.

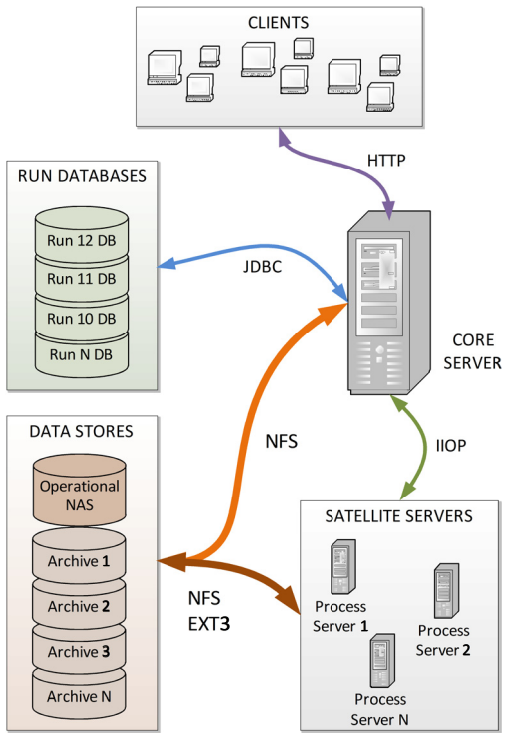


Figure 2: Data server communication diagram.

RESULTS

Data server's parallel processing core is only as fast as the backend, which supplies it the raw data. This is a non-issue for the cached values, however reading from a network store over a 1Gb/s connection can cause a significant bottleneck in the processing pipeline. Before any further development could be done, we had to prove to ourselves that the parallel design could indeed achieve the desired throughput rates. We also wanted to see how the rates scale for both compressed and uncompressed datasets. Figure 3 shows the effective data rates for both types of datasets with varying simultaneous thread count.

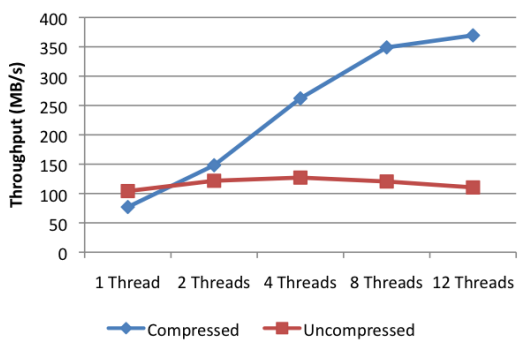


Figure 3: Effective Throughput Read Rates.

The uncompressed data reads over a network are essentially constant around 120MB/s and are unaffected by the additional threads. The same cannot be said for the compressed read scenario. The effective throughput on a 12-core Xeon server increases from 75MB/s to 365MB/s in a fairly linear fashion when moving from 1 to 12 threads. Results for the concurrent compressed scenario were encouraging enough to proceed with the project.

Performance Tests

Our most basic goal for this project was to improve the performance of displaying very large datasets. Our existing software for displaying logged data reads the data directly from disk files and displays the results. Data culling, as described above, is used when necessary to improve display performance. For comparison purposes, this software was modified to allow the option to retrieve data from the data server.

A variety of logged data was tested including very high frequency (10 kHz) and low frequency (1 Hz) data. To eliminate disk-caching issues, a 4 GB dataset was used to flush the cache in between each reading. In addition to having the data server read data from disk files, we also explored the option of reading data from a SSD data cache and from a RAM data cache. The results of these tests are shown in Table 1, which compares the speed at which the displayed data is transported through the various configurations.

Table 1: Data Throughput in Test Configurations

	Throughput	Speedup
Client to Remote Disk-store	5.4 MB/s	-
Client through data server to Remote Disk-store	146 MB/s	27x
Client through data server to SSD cache	245 MB/s	45x
Client through data server to RAM	968 MB/s	180x

The large speed advantage of using the data server (in the worst case, a factor of 27x improvement) is the result of the following:

- Network - the data server has a 1 Gb/s connection to the data file vs. a 100 Mb/s connection for the client.
- Parallel Execution - the data server has up to 12 threads that can individually read and cull data files. The client is single-threaded.
- Hardware - the data server has very fast CPUs with lots of RAM in a 64-bit environment. Clients are typically run on much more modest hardware.

The CPU speed turned out to be more important than we originally realized. This is because the data files are gzip-compressed files and need to be uncompressed before the proper data can be extracted. Fast CPUs significantly increase the speed of data decompression.

The results, while reproducible for a specific set of data, were quite variable across different data requests. In fact, the standard deviation across all of the speed tests within each category was about 50%. We have identified several factors that contribute to this variability:

- Data Density - Data for many items are stored together in files, primarily based on how users have setup logging requests. Extracting one or many from

the same set of files can significantly affect read performance.

- Timestamp Density - For slow data, we store one timestamp per data point. For most fast data we store data as arrays with a single timestamp and an indication of the time between points.
- Data Compression - Some data compresses much better than others. We've seen data compression rates range from 3 to 30 with 5 being a typical value. More highly compressed data can be read faster from disk because it is smaller.

SUMMARY

At this point it appears that our initial goal of decreasing the time it takes to display large datasets by a factor of 10 will easily be met by introducing a data server between client and data. Much of the speed gain has come from a fast network connection between the data server and the data, fast CPUs on the data server that can be run in parallel to process the data, and a culling algorithm that makes it possible to quickly transport data to the client.

There are several items that we have yet had time to explore. First, we would like to increase the network speed between the data server and our data files. Currently, they are connected via a 1 Gb/s network. We are exploring both 4 Gb/s and 10 Gb/s connections. Second, we would like to make a variety of data filters available to our users so that they can more quickly identify their data of interest. Filter types include "only data when the RHIC collider is on its energy ramp", "only data when there is beam in RHIC", and "only data when a measurement is above/below a threshold value". These filter selections will be passed down to the data server and reduce the amount of data that the server needs to process and the associated data that the user will view. Finally, we have plans on making better use of the solid-state drive connected to our data server. Past experiments have shown that users spend about 80% of their time looking at logged data collected within the last week and 90% looking at data collected within the last month. Our plan is to store recent logged data on our 1 TB SSD so that these frequent requests can be delivered more quickly.

REFERENCES

[1] K. Furukawa, M. Satoh, I. Mejuev, K. Nakao, "A Java-Based EPICS Archive Viewer With Soap Interface For Data Retrieval", http://accelconf.web.cern.ch/AccelConf/ica03/PAPERS/MP707.PDF, ICALEPCS (2003).

[2] Roderick, C., UK Oracle Users Group Conference, http://lhc-logging.web.cern.ch/lhc-logging/docs/Presentations/LHC_Logging_Service_UKOUG_2006.ppt (2006)

[3] Oracle Corporation, "Glassfish Metro Users Guide", http://metro.java.net/guide/

[4] Oracle Corporation, "JSR-000316 Java Platform, Enterprise Edition 6 Specification 6.0 Public Review Draft", http://download.oracle.com/otndocs/jcp/javaee-6.0-pr-oth-JSpec/

THE COMPUTING MODEL OF THE EXPERIMENTS AT PETRA III

Melvin Alfaro, Martin Flemming, Julia Grabitz, Thorsten Kracht*, Birgit Lewendel, Teresa Núñez, Peter van der Reest, André Rothkirch, Frank Schlünzen, Eugen Wintersberger
DESY, Hamburg, Germany

Abstract

The PETRA storage ring at DESY in Hamburg has been refurbished to become a highly brilliant synchrotron radiation source (now named PETRA III [1]). In comparison with the DORIS beamlines, the PETRA III experiments have larger complexity, higher data rates and require an integrated system for data storage and archiving, data processing and data distribution. Tango [2] and Sardana [3] are the main components of our online control system. Tango serves as the backbone to operate all beamline components, certain storage ring devices and equipment from our users. Sardana is an abstraction layer on top of Tango. It standardizes the hardware access, organizes experimental procedures, has a command line interface and provides widgets for graphical user interfaces.

The high brilliance together with the rapid read-out of modern 2D detectors dramatically increase the data rates and volumes. At PETRA III all data are transfered to an online file server which is hosted by the DESY computer center. Near real time analysis and reconstruction steps are executed on a CPU farm. A portal for remote data access is in preparation. Data archiving is done by dCache [4]. An offline file server has been installed for further analysis and inhouse data storage.

INTRODUCTION

By the year 2009 the reconstruction work to turn the storage ring PETRA into a highly-brilliant 3rd generation synchrotron radiation source was completed. The new facility, PETRA III, consists of 14 beamlines which are operated by DESY, the EMBL and HZG. The EMBL stations are independent of DESY as far as IT is concerned. By now, most of the beamlines are in user operation, the rest are currently commissioned.

DESY has a long-lasting experience in the field of synchrotron radiation experiment control which comes from the beamlines at DORIS, FLASH and PETRA. PETRA III creates new challenges. The beamlines have longer extent, consist of more elements and use a greater variety of electronic devices. In addition, the requirements of user interfaces are increasingly demanding. This applies to the scripting language, the command line and the graphical user interfaces. Consequently the online control system had to be newly designed. The original solution of controlling the experiments by a single program, Online [5], was replaced by a distributed client-server system.

The new 2D detectors which are used at PETRA III generate very high data rates. Therefore a concept for data acquisition, data management, reconstruction, analysis and data transfer had to be developed.

This note describes the layout of the computing model which has been implemented to meet the above mentioned requirements.

Figure 1: The PETRA III Hall.

EXPERIMENT CONTROL

The following issues have to be considered when designing an experiment control system:

- **Sustainability** A modular system with defined interfaces for hardware, communication and application layers can be extended and maintained by a group of software developers.
- **Flexibility** An efficient customization of the system is important for the best use of the beamtime. This customization is required due to the frequent change of user groups at the beamlines.
- **Platform independence** Linux and Windows have to be supported. Bindings to scripting and programming languages are mandatory.
- **Performance** An online control system needs sufficient bandwidth to handle motors, counters, timer, etc. However 2D detectors, creating by far the highest data rates, do not use the control system for data transfer but access storage media directly.

An online control system has to be based on a product that is supported by an international community. One reason is to save resources by benefiting from the work of others. At least as important is the communication among the members of the collaboration about new ideas and experiences. And finally, collaborations help to develop common user interfaces.

* Thorsten.Kracht@desy.de

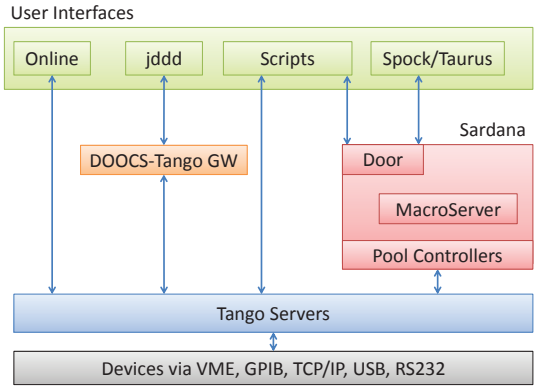


Figure 2: The main components of the experiment control system.

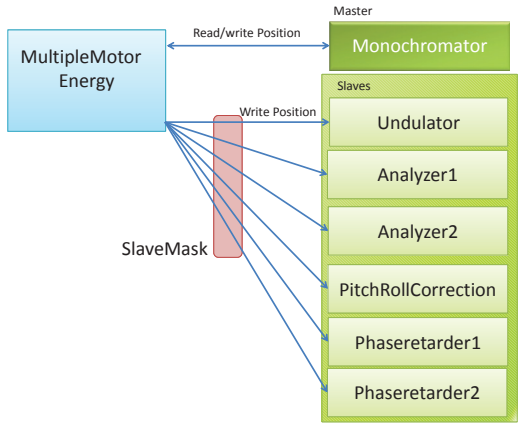


Figure 3: An example for a MultipleMotor server: the beamline energy.

Figure 2 gives a schematic view of the experiment control system. Tango serves as the hardware access layer. Clients can communicate directly to the Tango device servers or make use of the additional functionality that is provided by the Sardana framework.

The Tango Implementation

In order to limit the effect of corrupted database servers at least one Tango instance is installed for every beamline. Another choice was to group devices into classes with identical Tango interfaces. For cameras the LIMA [6] framework has been imported. Currently, area detectors available at PETRA III beamlines are being adopted.

The experiments at PETRA III use a variety of motors with many different parameters. It has been decided to implement a minimal interface in all motor classes, the TANGO_MOTOR. It executes the commands Calibrate, StopMove, ResetMotor and has the attributes Position, UnitLimitMin, UnitLimitMax. At DESY several motor servers of common interest have been developed. All of them implement the TANGO_MOTOR interface:

- **MultipleMotor** This server moves several TANGO_MOTORs concurrently. One of them is assigned to be the master which defines the position of the compound device. The slave motors, which follow the motion of the master, are enabled by a mask attribute, see figure 3.
- **AttributeMotor** A server that converts an attribute of a arbitrary device into a TANGO_MOTOR.
- **TcpIpMotor** This server uses an ASCII protocol via TCP/IP to control arbitraty motors that are temporarily installed by user groups.

Other general purpose servers that have been developed for PETRA III are: the CollisionSensor which avoids hardware damage due to unintentional movements and the ClipBoard server which facilitates a client-client communication by exporting several string attributes.

Tango Clients

These clients are currently used at the PETRA III experiments:

- **Online:** For a long time many DESY beamlines have been operated by this program. It has a command line interface, a binding to Perl and a graphical user interface.
- **Python scripts** Scientists started to write small python applications using PyTango [7] shortly after the PETRA III operation began. PyTango is a python module which exports the complete c++ Tango Api to Python.
- **Diffractometer Server** At PETRA III two beamlines have Eulerian 6-circle diffractometer. The diffractometer server, written by F. Picca of SOLEIL, is used for crystal orientation and direct-reciprocal space transformations imposing certain conditions.
- **jddd** jddd [8] applications exist for every beamline. They display selected variables for a quick overview of the setup, figure 4. jddd has a hierarchical structure with several expansion levels.

Although PETRA III started operation successfully using the above mentioned clients, it was planned to upgrade our user environment. We have chosen the Sardana framework to be the backbone of our future online control system.

Sardana at PETRA III

Sardana was created at ALBA. Important contributions came from the ESRF. Sardana is a framework that structures the domain of experiment control programs and user interfaces. It has several components: the device pool is an additional hardware standardization layer, the MacroServer organizes the execution of procedures that are needed for

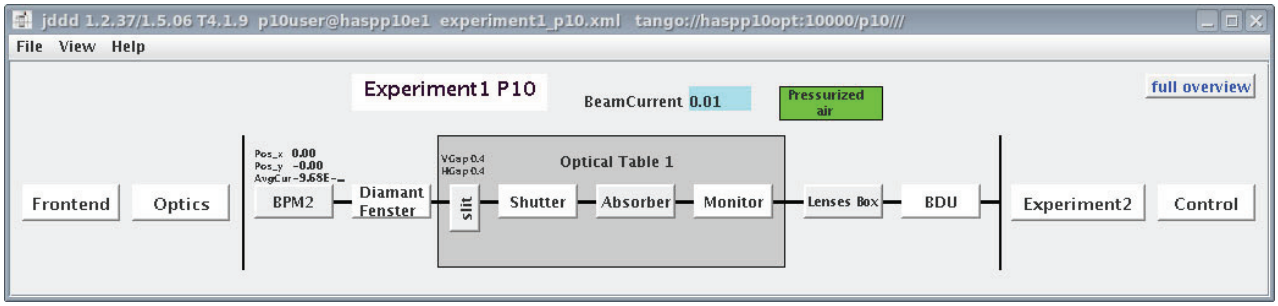


Figure 4: jddd application of a PETRA III beamline, partially expanded.

the measurements, Spock serves as the command line interface and Taurus builds GUIs. These features make Sardana well suited for PETRA III. In the meantime DESY participated in this project by adding some extensions. It has been rolled-out recently to the experiment PCs. First test measurements have been performed.

Data Format

The current practice to produce in a single experiment myriads of small files in a large variety of partially proprietary formats has a bad impact on the performance of the storage and archive system, tremendously increases the complexity of the data management tasks, hinders application development, prevents usage of data across disciplines as well as facilities and makes the long-term preservation of scientific data almost impossible.

To enable users to work with data collected at different beamlines at the same or different facilities, the PNI-HDRI [9] initiative (in close co-operation with the PaN-data project [10]) aims to develop and implement a standard data format, which also permits to aggregate data files from different sources into a performant, portable and self-describing form.

Nexus [11] has been selected as the underlying file format since it's based itself on widely accepted standards (HDF5) and because it has a growing user community in the photon, ion and neutron science world. It is part of the HDRI project to demonstrate experiment specific implementations and to provide guidelines how the application specific information is organized. The experiments at PETRA III will follow the developments of HDRI.

It cannot be expected that the Nexus will generally be accepted by the international photon science community. SOLEIL and ANSTO proposed to add an abstraction layer, CDMA [12], to help application developers to uniformly access files. CDMA uses a dictionary mechanism to translate application notions to data file entities. In order to benefit from these developments, it is planned to write a CDMA plugin for Nexus.

DATA MANAGEMENT

So far it was discussed how an experimental setup is controlled to perform measurements and collect data. This section describes how this data is treated afterwards.

The PETRA III beamlines send their output directly to the online file server. It has the purpose to buffer the data before they are archived and to make them available for near real time analysis, figure 5. All data are copied to the dCache tape pool for archiving. Data to be analyzed is staged to the dCache disk pool. In addition there is an offline file server for storing analysis results and temporary data. These services are provided by the DESY computer center.

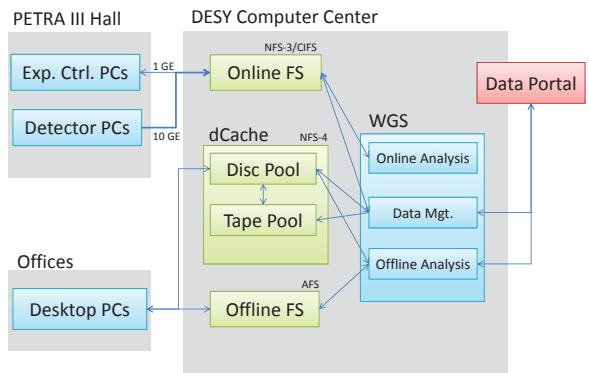


Figure 5: Main data and control flow paths.

Network

Internally the PETRA III experiment hall has a 1 GE Ethernet infrastructure. The hall itself is connected to the DESY computer center by 10 GE Ethernet. Detectors which generate high data rates are connected by 10 GE lines. The effective speed for writing data to disk via the 10 GE lines is about 300 MB/s (from Linux PCs).

File Servers

The online file server storing the data during measurements can be accessed from either Linux or Windows™ systems. Typically, the Linux based experiment PCs and the detector PCs mount this server by NFS

(version 3). It offers CIFS-shares for Windows™ based systems.

The online file server consists of 2 file server systems, each managing 11 drive trays of 14 disks each. It is set up as a raid 6 storage system, has takeover options for the 2 heads in a single system and is operated with snapshot option of 20% of total storage. The total capacity of the online file server is 145 TB for all beamlines (excluding snapshot and spare disks).

If an experiment needs a fast reconstruction step to optimize the measurement, it is done with the data on the online file server.

The offline file server stores analysis results, no raw data. It is accessible from the experiment PCs and the workgroup server via AFS. The capacity of this file server is about 40 TB in a raid 6 setup.

dCache

The dCache is a storage system consisting of a disk pool and a tape pool. It was developed by DESY and FERMILAB for storing high energy physics data. The dCache becomes increasingly important for the photon science department of DESY.

Data are migrated from the online file server to the dCache. During this migration the ownership changes from the beamline account to a user-specific account. The beamline scientists in charge keep full access rights.

Data are directly accessible (via NFS4.1 exports) by analysis programs as long as they are stored on the disk pool. Data that are already on the tape pool need to be staged before they can be read. Data stay on the disk pool for at most 180 days. The residence time in the tape pool is not limited so far.

Data Portal

It is expected that certain guest groups do not have the sufficient IT infrastructure at their home institutes to store and process large data volumes. DESY plans to provide these scientists with storage and compute services. A data portal is currently being prepared that facilities access to the dCache and permits the full remote management of the data, which includes the search for data and meta-data and a user controlled rights management.

Compute Server

Currently PETRA III computing is performed on a blade center which is divided into 16 work group servers (WGS). They have a dual-quad-core cpu with 24 GB RAM and 16 GB swap space running 64bit Scientific Linux. Part of the WGS are organized in a server pool, the remaining ones are addressed directly and can be allocated to specific beamlines. Computing can also be performed on Nvidia™ GPUs. WGS and GPUs are used for reconstruction, analysis and simulation as well as for software compatibility tests or data migration.

CONCLUSIONS

PETRA III computing integrates products that originate from activities of international and national partners: Tango, Sardana, CDMA, dCache. This approach saves resources and enhances the communication on the relevant topics which is crucial for the fast and continous incorporation of new developments. Furthermore, collaborations among synchrotron radiation facilities are the precondition for establishing common user interfaces. A feat that would ease the burden of our scientific users in performing measurements at different sources.

It has been demonstrated that the experiment control system consists of several layers with well defined interfaces making it flexible enough to fulfill the current requirements and to cope with future challenges.

ACKNOWLEDGMENTS

We would like to thank all who made contributions to the Tango control system and our colleagues from ALBA for having created Sardana, in particular T. Coutinho, C. Pascual and J. Klora who gave us so much support. We thank F. Picca for allowing us to use his diffractometer server. We also acknowledge the work of DESY-MCS on jddd.

REFERENCES

[1] The New Synchrotron Radiation Source at DESY, http://petra3-project.desy.de.

[2] Tango Control System http://www.tango-controls.org.

[3] J. Klora, T. Coutinho et al., The architecture of the ALBA Control System, Proceedings NOBUGS 2008

[4] http://www-dcache.desy.de.

[5] Online, Data Acquisition and Beamline Control, http://hasylab.desy.de/online.

[6] LIMA, Library for Image Acquisition, http://forge.epn-campus.eu/Lima.

[7] PyTango, http://www.tango-controls.org/static/PyTango/latest/doc/html/index.html.

[8] jddd, A Java DOOCS Data Display, http://jddd.desy.de

[9] High Data Rate Initiative for Photons, Neutron and Ions, http://www.pni-hdri.de/.

[10] PANDATA – Photon and Neutron Data Infrastructure, http://www.pan-data.eu/.

[11] NeXus, http://www.nexusformat.org.

[12] N. Hauser et al., CommonDataModel. A Unified Layer to Access Data from Data Analysis Applications Point of View, proceedings, ICALEPCS 2011

DATABASE FOUNDATION FOR THE CONFIGURATION MANAGEMENT OF THE CERN ACCELERATOR CONTROLS SYSTEMS

Z. Zaharieva, M. Martin Marquez, M. Peryt, CERN, Geneva, Switzerland

Abstract

The Controls Configuration Database (CCDB) and its interfaces have been developed over the last 25 years in order to become nowadays the basis for the Configuration Management of the Controls System for all accelerators at CERN.

The CCDB contains data for all configuration items and their relationships, required for the correct functioning of the Controls System. The configuration items are quite heterogeneous, depicting different areas of the Controls System – ranging from 3000 Front-End Computers, 75 000 software devices allowing remote control of the accelerators, to valid states of the Accelerators Timing System.

The article will describe the different areas of the CCDB, their interdependencies and the challenges to establish the data model for such a diverse configuration management database, serving a multitude of clients.

The CCDB tracks the life of the configuration items by allowing their clear identification, triggering of change management processes as well as providing status accounting and audits. This necessitated the development and implementation of a combination of tailored processes and tools.

The Controls System is a data-driven one - the data stored in the CCDB is extracted and propagated to the controls hardware in order to configure it remotely. Therefore a special attention is placed on data security and data integrity as an incorrectly configured item can have a direct impact on the operation of the accelerators.

INTRODUCTION

The idea to use a central data storage to describe the components of the Controls System for CERN's Proton Sychrotron (PS) complex was suggested in 1980. Several implementations were made using a variety of database models such as an object oriented one. In 1986 an implementation using a relational database model and in particular using the Oracle® Relational Database Management System (RDBMS) was provided. This marked the birth of the Controls Configuration Database, which has been in service ever since [1]. Throughout its existence it has been constantly evolving, growing in size and in provided functionality in order to cover the increasing complexity of the accelerators Controls System at CERN.

THE NEED FOR CONFIGURATION MANAGEMENT

Configuration Management was established as an important element of Systems Engineering during the 1950s and 1960s. Realizing its importance not only for the hardware configuration of systems, but also for software configuration, was the reason to include it as a component of best practices, formalized by Information Technologies Infrastructure Library (ITIL®) [2].

The amount of technical data, necessary for the control of the CERN accelerator complex is enormous. A common description, in a centralized storage, of all objects needed for the Controls of the accelerators is an essential prerequisite for the correct and coherent functioning of the accelerators.

The CCDB and its interfaces provide configuration management functionalities such as the unique identification and configuration of items, complying with predefined criteria, as well as controlling the configuration changes and status accounting of the configured items. Those functionalities answer the need to establish and maintain the consistency of the Controls System and its components.

SCOPE

The CCDB is the heart of the CERN Accelerators Controls System. Nowadays it covers the need for the configuration of components of the Controls System itself, e.g. the Controls Middleware (CMW), as well as accelerator components as seen by the Controls System, e.g. power converters, for all accelerators: the Large Hadron Collider (LHC), the Super Proton Synchrotron (SPS) Complex, the Proton Synchrotron (PS) Complex, and the CLIC Test Facility (CTF3). The configurations of some components used for the Control Systems for the Technical Infrastructure services at CERN are done in the CCDB too.

Providing the configuration capabilities for such a diverse number of systems and components and representing them in a unified model is quite a challenge. Currently the relational database model of the CCDB comprises of 914 tables, storing over 10GB of current reference data and 50GB of historical versioned data.

CONFIGURATION MANAGEMENT FUNCTIONALITIES, PROVIDED BY THE CCDB AND ASSOCIATED INTERFACES

Configuration of all Components of the Controls System

The data in the CCDB represents components and their properties as seen by the Controls System. It contains data for the complete Controls System topology with some of the main users of the CCDB services being the CMW, the Role Based Access (RBAC), the Accelerator Timing System, the Common Console Manager, etc. (see Fig.1).

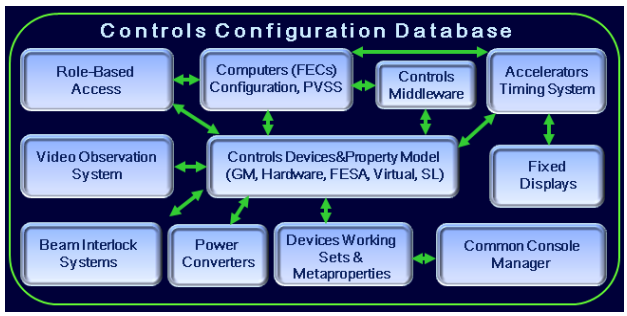


Figure 1: Overview of the different areas of the CCDB.

The Controls devices are the software representations of the accelerator components, controlled by the Controls System, e.g. power converters, radio frequency (RF), beam instrumentation (BI) components, etc. The CCDB currently supports 5 device-property models – General Module (GM), Front-End Software Architecture (FESA), SPS&LEP (SL), Hardware and Virtual. There are more than 1,400 device classes, more than 79,000 devices and more than 2,000,000 properties declared in the CCDB.

The GM and SL legacy frameworks are in the process of being phased-out; they were developed respectively for the PS and SPS complexes.

The FESA framework was introduced 8 years ago. The original data management solution was to store directly XML configuration files in the CCDB, produced by the FESA development tools. This solution presented several shortcomings mainly with the integration of the FESA-related data to the rest of the configuration data in the CCDB. Another problem was tracking of devices and properties changes, contained within the XML files. A major effort to restructure the data management part of the new FESA framework version (3.0) was initiated in 2009 [3]. The main challenge was to implement a structured handling of XML files, which should be used only for data exchange between the FESA end-user tools and the CCDB. The XML files are no longer stored in the CCDB - their contents are extracted on the fly (using a PL/SQL API) and the FESA configuration data is stored into the relational tables. This solution has allowed a complete integration of FESA data into the CCDB relational model and facilitates the handling of common data management tasks.

The Hardware devices framework was developed in 2007 and originally dedicated to the Power Converter Controls [4] devices and PVSS devices for the Machine Protection of LHC (Quench Protection System, Warm Interlock System, Power Interlock System, etc.). Nowadays the scope of this framework has been broadened to a wide variety of physical equipment.

The Virtual devices framework was the last Controls devices framework to be introduced in the CCDB (only 1 year ago) and represents abstract elements that need to be controlled, e.g. signals for the Software Interlock System.

The challenge before the CCDB was to implement as much as possible a unified database model, which suites those 5 diverse Controls devices frameworks [5]. It is the database that provides a layer of abstraction and provides the device-property data in one and the same way to the multiple db clients (e.g. CMW, RBAC).

The CCDB also provides data for the hardware and software configuration (e.g. start-up sequences) of all Front-End computers used by the Controls System. There are more than 3,200 computers declared in the CCDB. Special functionality is available to configure the drivers for the modules, used in the Front-End computers.

The Accelerators Timing System configuration is another important area. At the beginning of 2011 a major renovation of the Timing System was undertaken by the timing experts and this had an impact on the CCDB. The database model was completely reworked in order to introduce high-level timing events and to allow full configuration of low-level timing events. The new model ensures the completeness of the configuration of the Timing System. It allows better integration of the timing data with the rest of the configuration items in CCDB.

It is important to have a complete description of the components, needed for the accelerators Control, stored in a single location such as the CCDB. This provides the possibility to validate configurations against business rules and to establish a coherent picture of all components and their relationships (dependencies between the components).

Extraction of Configurations – Data-Driven Controls System

The CCDB is not merely a storage repository for the configuration of the different components, related to the Controls of the accelerator. The CCDB actively serves to provide real-time configuration data, which is extracted from the database whenever a component needs to be configured. In order to provide the on-line data for the configuration items and their relationships, there are a number of APIs and scripts which have been developed.

Configuration Change Management

For the data-driven Controls System the data in the CCDB describes the Controls components and their properties. Those components need to be upgraded regularly, e.g. Front-End Computers, Device Classes, etc, therefore there is a need for regular data modifications and maintenance in the CCDB.

Each configuration item in CCDB is uniquely identified, which allows the database to trace the history of all operations (data modifications) performed on that item. The users are provided with reports on the current configuration and its evolution over time.

A solution to review potential configuration changes and accept or reject them is under development. The change acceptance process is mainly guided by the status of a given component (operational or not) and whether a change is considered to be backward compatible [6].

On-line Feedback of Deployed Configurations

A recently implemented new feature of the Controls Configuration is the availability of status accounting of the different configuration items. The status data comes in

the form of a feedback from the configured components identifying the currently loaded configuration.

The first configuration items to start sending feedback data are the drivers for the Front-End computer modules and some components from the CMW configurations. This feature of the Controls Configuration Management is extremely important as it solves the years-old problems of discrepancies between the loaded Front-End configurations and the current running configurations, resulting in an unexpected behaviour after a reboot.

It is previewed that this functionality will be extended to cover other areas of the Controls Configuration in the near future, e.g. the software devices configurations deployed on the Front-Ends, etc.

OVERVIEW OF THE CONTROLS CONFIGURATION ENVIRONMENT

Database Complexity

The Controls Configuration database is quite a complex one as it models the Controls System into a relational database. The main challenge for the database model is to maintain data consistency and to enforce the predefined business rules. This is achieved by numerous constraints as well as thousands of lines of PL/SQL code in database triggers or packages. Table 1 provides the exact figures showing the complexity of the CCDB data model.

Table 1: CCDB Statistics

Tables	914
Constraints	2,388
Lines PL/SQL code	42,100
Volume	60GB

Database High Availability

The Controls Configuration services need to be continuously available 24 hours a day, 365 days a year to assure the ability to configure components at any moment in order to operate the various accelerators at CERN as well as to allow configuration changes to be introduced during scheduled accelerator "technical stops". This has been achieved through the use of Oracle cluster technology (RAC) which guarantees not only the hardware but also database software redundancy. This solution ensures no down time even during routine software patching of the database.

Configuration Data Responsibility

There is a diverse user community for the Controls Configuration services, which makes the job of providing tailored user applications challenging. The responsibility to maintain the correctness of the configuration data lies with the relevant equipment experts (BI, RF, etc.), as well as controls experts and accelerator operators.

Data Editing Interfaces

The CCDB Data Editing Interfaces are a suite of web-deployed applications used to modify the stored configuration data. There are currently 12 Configuration Editors, which implement the business logic and processes in the various areas of the CCDB. These applications are based on Oracle's J2EE ADF technology and are used on a daily basis by more than 250 users.

Special attention must be paid to the access rights given to the users of these applications, which need to be tightly controlled due to the sensitive nature of the data and the associated risk of mis-configuration of accelerator components. Strict access control rules are therefore implemented via a custom authorization module. Through the use of the virtual private database feature of Oracle, additional fine grained access control mechanisms are provided within each application.

Data Browsing Interfaces

The CCDB Data Browsing Interfaces comprise of read-only applications (reporting tools), for situations where the configuration data only needs to be consulted. Some 160 reports covering all areas of the CCDB are provided to the user community of roughly 300 people and are built using Oracle APEX technology [7].

APIs and Scripts

Various APIs, written in several languages such as Java (e.g. Java Directory Service, Beam Interlock Systems API), PL/SQL (e.g. Front-Ends Drivers Generation APIs, FESA data extraction API) and some legacy Pro*C scripts (e.g. Front-End configurations), are implemented to extract the configuration data or to generate files to be used by the different components of the Controls System.

Diverse output formats of the components configurations are produced from the CCDB, ranging from text files for drivers generation and hardware and software configuration of computers to XML files and binaries for Controls devices configuration.

DATA SECURITY

Data security is paramount in the CCDB. The CCDB provides specially developed functionality to audit every session opened in the CCDB, allowing monitoring of who changed what data and when.

In order to trace data changes, a custom history framework was developed. All data modifications are recorded and kept on-line since 2005. There is a special web-deployed application – the CCDB History Browser, which gives access to the history logs. This application is used heavily by the Controls Exploitation team.

The history log also serves as a basis for versioning of the configuration data for each configuration item.

QUALITY ASSURANCE – TESTING AND DEVELOPMENT ENVIRONMENTS

The Control of CERN's accelerators is a very dynamic environment with new and changing user requirements coming regularly, directly impacting the database model as well as the different applications and APIs. The best software development style to suit such an environment is

agile programming, facilitating fast prototyping and short time to production to achieve a feature complete solution.

Four environments have been provided for the CCDB, its interfaces and APIs – *Development*, *Test* – used for unit and functional testing, *Next* – used for integration testing and *Production* environments.

The Next environment of the Controls Configuration is part of the Controls TestBed since 2010 [8]. Some changes in the Controls components are so fundamental, e.g. the renovation of the accelerators Timing System in 2011, that it could take more than 1 year before deploying the new functionality to production. The TestBed provides unique opportunities to perform integration and system testing of the new functionalities with all major components of the Controls System being available.

DATA PROPAGATION

The data management for the Control and Operation of CERN's accelerators is implemented as a distributed database environment [9]. Part of the data stored in the CCDB is propagated to other database systems for the needs of the Operation of the accelerators (see Fig.2).

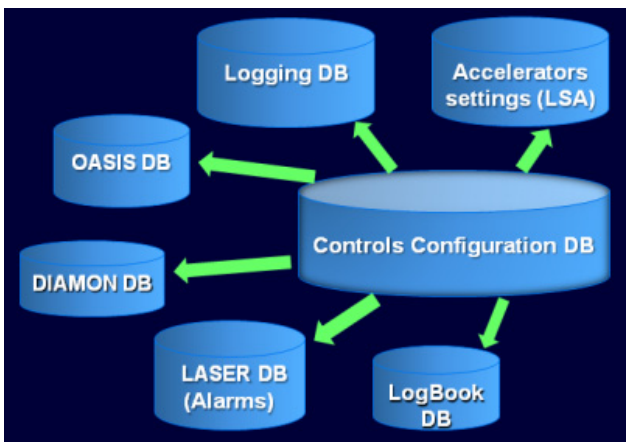


Figure 2: Propagation of data from the CCDB to other databases, used for the Operation of the accelerators.

The alarm definitions (>120,000) for all accelerators devices, are generated and propagated to the Alarms Database [10]. Part of the devices' data is propagated towards the Accelerators Settings (LSA) database, where it is used as a basis for the Operation of the accelerators. The CCDB provides the data for the computers (Front-Ends, consoles, servers, etc.), which need to be monitored by the Diagnostics and Monitoring System (DIAMON).

Safe Propagation of Data Changes

Changes to configuration data in CCDB could have an impact on the related databases. A strategy for smooth data upgrades of the data-driven Controls System has been established with the objective to ensure a coherent set of data throughout all distributed databases. Database procedures are developed in the CCDB to safely propagate data changes, based on knowing the data dependencies between the different systems. The impact of the configuration data changes is analyzed and appropriate actions taken – this process is part of the configuration Change Management functionality provided by the CCDB.

CONCLUSION

The Configuration Management has proven to be an indispensable part of ensuring the correct functioning of any large system.

The Controls Configuration DB, its interfaces and the specific processes implemented around those, are providing the basis for the Configuration Management of the Controls System for all accelerators at CERN. The CCDB ensures conceptual unification and centralization of the diverse configurations of the different items and their relations, thus describing the different components of the Controls System and their dependencies.

Continuous effort is being put into rationalizing, improving, federating and developing new functionality in the existing database and its interfaces.

REFERENCES

[1] J. Cuperus, R. Billen and M. Lelaizant, "The Configuration Database for the CERN Accelerator Control System", ICALEPCS'03, Gyeongju, Korea, Oct-2003, WE114.

[2] ITIL®: www.itil-officialsite.com

[3] M. Arruat *et al.*, "FESA 3.0", ICALEPCS'11, Grenoble, France, Oct-2011, WEPMN002.

[4] Q. King, S.T. Page and Z. Zaharieva, "Automatic Inventory and Configuration Management Tools for the LHC Power Converter Controls", ICALEPCS'09, Kobe, Japan, Oct-2009, TUA004.

[5] M.Peryt *et al.*, "Database and interface modifications: change management without affecting the clients", ICALEPCS'11, Grenoble, France, Oct-2011, MOPKN010.

[6] V.Baggiolini *et al.*, "Backward Compatibility as a Key Measure for Smooth Upgrades to the LHC control system", ICALEPCS'11, Grenoble, France, Oct-2011, WEPMS007.

[7] Z. Zaharieva, R. Billen, "Rapid Development of Database Interfaces with Oracle APEX, used for the Controls Systems at CERN", ICALEPCS'09, Kobe, Japan, Oct-2009, THP108.

[8] J.Nguyen Xuan, V.Baggiolini, "Testbed for validating the LHC controls system core before deployment", ICALEPCS'11, Grenoble, France, Oct-2011, WEPMS003

[9] R. Billen *et al.*, "Accelerator Data Foundation: How It All Fits Together", ICALEPCS'09, Kobe, Japan, Oct-2009, TUB001.

[10] Z. Zaharieva, M. Buttner, "CERN Alarms data management: state & improvements", ICALEPCS'11, Grenoble, France, Oct-2011, MOPKN011

INTEGRATED APPROACH TO THE DEVELOPMENT OF THE ITER CONTROL SYSTEM CONFIGURATION DATA

D. Stepanov, L. Abadie, ITER Organization, Route de Vinon sur Verdon, 13115 Saint Paul Lez Durance, France

J. Bertin, G. Bourguignon, G. Darcourt, Sopra Group, ZAC de Pichaury II - 780, rue Guilibert de la Lauzière - BP 25000, FR 13791, Aix-en-Provence cedex 3, France

O. Liotard, Tata Consultancy Services, La Défense 8, 100-101 Quartier Boieldieu, 2800 Puteaux France

Abstract

ITER control system will rely on a large number of configuration data, coming from different sources. This information is being created using different tools, stored in various databases and, generally, has different lifecycle. In many cases it is difficult for instrumentation and control (I&C) engineers to have a common view on this information or to check data consistency. The plant system profile database, described in this paper, tries to address these issues by gathering all I&C-specific information in the same database and providing means to analyze these data.

INTRODUCTION

ITER control system, CODAC, has a major challenge of not being created by a single team in a single location, but instead split into different pieces according to the plant systems manufacturing and delivery process. This fact increases substantially the number of people involved in the I&C design, which leads to many different practices and approaches to the implementation of I&C systems. The CODAC team takes preventive measures to reduce diversity by standardizing procedures, hardware and software via its Plant Control Design Handbook (PCDH) and the software product called CODAC Core System (see [1]). The CODAC Core System [2] is a scaled down version of future CODAC, based on EPICS [3], providing essential software support for creating locally of a control system "island" of an arbitrary complexity. These pieces of the control system are called "plant system I&Cs" in ITER terminology.

The configuration data developed with the help of the Core System is conventionally called the "self-description data" (SDD) of particular plant systems, because it allows configuring many elements of the central CODAC by just reading these data (see more on the ITER SDD concept in [4]). The Core System comes with its own relational database, based on PostgreSQL [5], and the tool, called the SDD editor, to create plant system I&C configuration using a top-down approach. This database-enabled solution was initially released in February 2011 and it proven itself being well adapted to I&C engineers' needs. It should be noted that, with the dissemination of Core System installations around the globe, the number of such databases grows, and their content has to be collected and integrated. The workflow to collect these configuration data into the central database is explained in [4].

While the Core System SDD tools are mostly concentrated on the control software configuration part, there are quite a few other areas in the ITER design activities which have impact on I&C configuration. The I&C relies heavily on naming convention for plant components and signals, which is defined and maintained project-wide. The I&C hardware has to be properly allocated in racks and cupboards ("cubicles" in ITER terminology), and those, in turn, have to be assigned to the right locations on the ITER site. The I&C architecture design process is supported with 2-D diagram tool, which has its own database for drawings and I&C objects. The cabling accounting will also be supported by a database solution. Finally, the progress of I&C manufacturing and procurement has to be monitored, and the procured equipment has to be properly registered and accounted. All these tasks are often solved with the help of the tools which are best in their class for functionality, but inevitably bring some diversity with databases and data schemas that accompany them. There is an on-going effort at ITER to unify all engineering data under the umbrella of so-called "engineering database", but the solution has not arrived to production level at the moment and is not yet much I&C-specific.

These circumstances lead to a natural idea of a "syndicated" I&C-specific database which is capable to collect all the I&C-relevant data in a single place and present it in a coherent way. This is what we call a "plant system profile database".

SCOPE AND OBJECTIVES

The main objective was to provide the ITER I&C team with a tool to address immediate needs for configuration data collection and analysis. The scope of the data itself is not always well defined and highly depends on current priorities of work. Thus we opted for a flexible approach, which consists of: 1) providing a solution generic enough to work with any kind of structured data; 2) approaching areas of interest step by step, by determining their properties and implementing them in agreement with the rest of the database.

The most immediate needs at the moment are capturing some top level I&C design decisions and quantitative estimates, as well as tracking the progress of I&C design

and procurement. Consequently, the scope of initial works was to handle the following data:

- breakdown of ITER into plant systems and plant system I&Cs;
- I&C estimates, like estimates of number of cubicles and signals;
- detailed lists of components, signals and I&C variables;
- tracking of procurement arrangements, design reviews, design deliverables, reference documentation.

The objective was attained, even though in many cases the data has not been entered in the database yet or simply does not exist at the moment. Details on the implementation are given in the next section. From this point we are looking forward to address more advanced topics, like:

- support of the remote CODAC Core System databases (a so-called "SDD repository");
- implementation of a workflow between the 2-D diagrams tool and CODAC databases;
- component life cycle management and inventory control;
- support for interlock functional analysis;
- automation of certain routine data imports and data quality checks.

FIRST IMPLEMENTATION

At the design phase, the current ITER infrastructure and CODAC practices have been evaluated, and the following architectural and software decisions have been made:

1) the database backend should be MicroSoft SQL Server [6];
2) the application should have web interface;
3) business logic and the user interface should be written in Java. PrimeFaces [7] should be used for the user interface, Spring [8] for transactional support and Java Hibernate [9] for interaction with the database;
4) data integration tool should be Talend [10].

The task was launched in September 2010; in February 2011 the first version was put in production, and it was gradually introduced into CODAC processes in the following months.

The front page of the application is presented on Fig. 1. It presents an overview of data stored in the database at the moment. The key metrics are number of plant systems and plant system I&Cs ("control islands"), the number of their components and signals, the numbers of I&C controllers and variables handled by them. As one can observe, the population of the database has just started. It is worth noting that the database is not only about technical data – it collects and presents information about administrative aspects of I&C tracking – like procurement arrangements, design review and documentation status. Since the administrative data is available, the database application is able to offer a list of coming milestones right at the front page.

Various aspects of data can be studied using so-called "perspectives" (the names comes from Eclipse perspectives, similar by nature). The perspectives are views of the same data presented from different angles. The main perspectives defined today are:

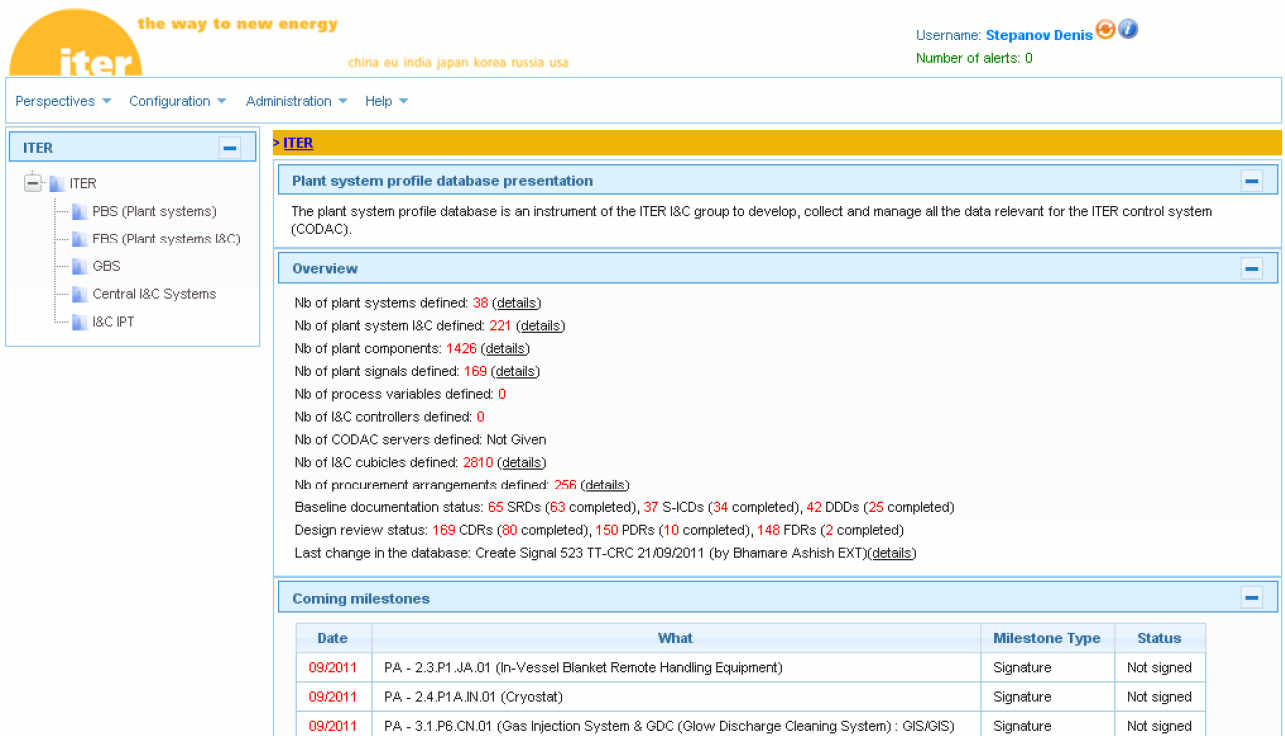


Figure 1: Plant system profile database entry page.

- PBS (Plant Breakdown Structure) perspective – seen from the point of view of plant systems;
- FBS (Functional Breakdown Structure) perspective – seen from the point of view of plant system controls;
- GBS (Geographical Breakdown Structure) perspective – seen from the geographical point of view;
- Central I&C systems perspective – view on configuration of central systems;
- I&C IPT (Integrated Product Team) perspective – organizational and administrative view.

The same object (e.g., a signal or a controller) can appear in various perspectives, with a focus on its properties valuable for that view.

The PBS perspective allows navigating through the list of plant systems and finding out which components, cubicles, signals belong to it and what are the buildings which host the plant system. It also allows entering quantitative estimations, the people responsible for the system and references to the relevant ITER baseline documentation.

The FBS perspective is focused on I&C and presents it as a tree of control functions. It is possible to see the list of functions, control units and variables belonging to a plant system I&C, or enter estimations of those. With the information entered, the database, in principle, can derive a drawing of the I&C architecture. A very simple example, based on the tokamak cooling water system specification, is shown on Fig. 2.

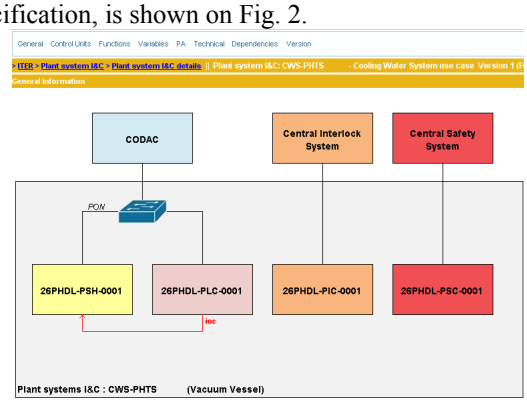


Figure 2: A sketch of a cooling water system I&C, generated by the database web application.

Here one can easily see that the plant system I&C has one slow controller (PLC) which is served by a plant system host running an EPICS Input/Output Controller (IOC). Interlock and safety controllers are equally declared. By clicking on the boxes, one can navigate to the definition of particular controller. The figures like this require zero effort from the end user and are ready to be used in various specification documents, or in presentations. One more advantage is that they use the same styles and PCDH methodology across all ITER systems.

The GBS perspective shows a map of the ITER site and allows going down to individual buildings and rooms and finding out the equipment, such as cubicles, planned to be installed there.

The central I&C systems perspective is focused on configuration of central systems. Currently it only shows the list of CODAC network nodes ("network hutches") which interconnect different buildings. There will be more information added on CODAC special purpose networks, servers, as well as interlock and safety systems.

Finally, the I&C IPT perspective represents an administrative view on the systems. There one can learn the organizational breakdown of people assigned to work on particular systems, information on procurement arrangements, tracking of milestones and documents. One of important activities at the moment is participation of the CODAC team in various design reviews of the ITER plant system systems to make sure that I&C procurement is not forgotten and is designed according to the ITER standards. The reviews are normally structured per procurement packages and pass through different levels of maturity (conceptual, preliminary, or final design). Given the number of procurement arrangements which involve I&C (235 registered in the database at the moment of writing), one can easily understand the amount of work load incurred, which has to be properly planned and accounted. The I&C IPT perspective helps to manage this work by carefully recording all the planned or completed reviews together with the associated documents and remarks from the team.

DATA ANALYSIS AND REPORTS

In many cases it is not sufficient just to collect the I&C data; it is equally important to be able to perform data analysis and present properly certain information of interest. Topics of particular interest for ITER at this moment include:

- Analysis of plant system I&C design in order to understand and have under control the number of key I&C elements – I&C controllers, I/O boards, cubicles and to see if they respond to the envelops defined, such as footprints in rooms and areas designated to host the I&C equipment;
- Analysis of performance requirements and identification of possible bottlenecks in processing data transfer, or response times which may lead to refinement of requirements to the CODAC System or even prompt redesign of some systems;
- Tracking of the I&C design and implementation progress and comparison it with the overall project schedule in order to identify schedule slippages and take preventive measures;
- Estimations of cost of the control system based on average values of key parameters, like a cost of a data acquisition channel.

Here, different implementation approaches could be taken. Key indicators which need to be accessed daily and instantly can be built right into the web application interface, like in the case of dashboard on Fig. 1. More powerful, but less integrated way is to use a reporting services mechanism (MS SQL Server Reporting services in our case), which provides rich presentation layer and

can be created relatively quickly on demand. This is a convenient way of managing reports which need to be accessed regularly. Finally, one can export data of interest into an Excel file and apply processing on his or her own, which enables full presentational power of modern office packages.

Below are some examples of reports created with the help of reporting services. Figure 3 shows a distribution of I&C cubicles among the buildings on the ITER site:

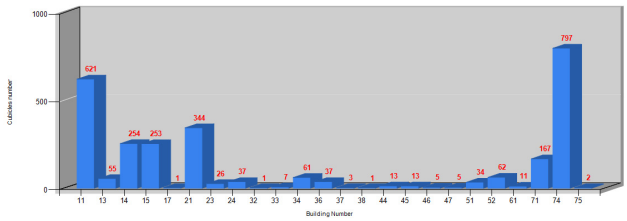


Figure 3: I&C cubicles broken down by buildings.

This graph allows grasping quickly which buildings will be the most I&C-loaded and thus will require particular investment into cabling and network infrastructure.

Another example (Fig. 4) shows a distribution of I&C-related procurement arrangements among the plant systems and the ITER domestic agencies.

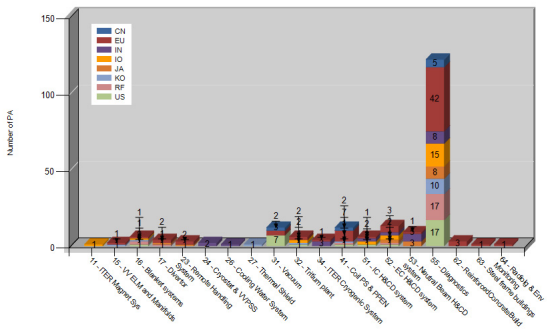


Figure 4: I&C procurement breakdown per plant system.

Finally, to support the I&C progress meetings, where the status of I&C procurement is reviewed and discussed, a specific dashboard was defined. This page, shown on Fig. 5, represents procurement organization, key performance indicators, deliverables and milestones for a given plant system and allows quick understanding whether the I&C procurement is on track and what are the particular issues that have to be resolved. The history of reports can be maintained, e.g., on a weekly basis, so one can go back in time at any given moment, or build a progress graph of key performance indicators evolving with time.

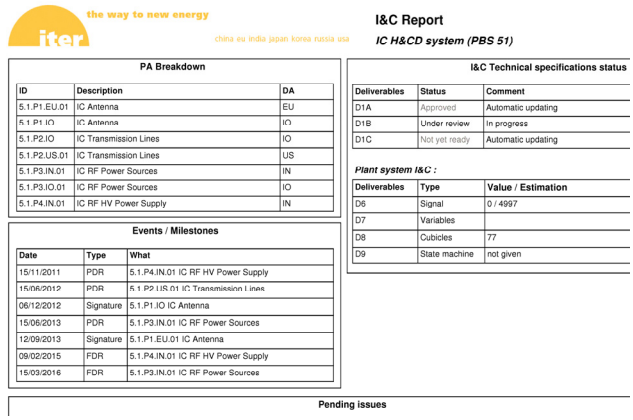


Figure 5: Example of the I&C design and procurement status for the ion-cyclotron heating system.

CONCLUSIONS

The extensive work done by the CODAC team to support the I&C design of the ITER plant systems highlighted the need to supplement this activity with a database application. The plant system profile database, put in production in 2011, has just barely started, but already allows filing the I&C design data in a unified way and calculating useful data metrics. The work is now focused to put the entire information collected so far under the control of the database and to continue integration activities with the CODAC control software and the rest of the ITER databases.

The authors wish to thank the CODAC team and the I&C IPT for their support and suggestions which helped to define and continue to steer the development of the plant system profile database.

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

[1] A. Wallander et al, News from ITER Control – a Status Report, this conference.
[2] F. Di Maio et al, The CODAC Software Distribution for the ITER Plant Systems, this conference.
[3] EPICS Control System, http://www.aps.anl.gov/epics/
[4] L. Abadie et al., "The self-description data configuration model", IAEA TM8, San Francisco, June 2011
[5] PostgreSQL database, http://www.postgresql.org/
[6] MicroSoft SQL Server, http://www.microsoft.com/sqlserver/
[7] PrimeFaces, a JSF implementation, http://www.primefaces.org/
[8] Spring Java framework, http://www.springsource.org/
[9] Java Hibernate, a relational persistency framework, http://www.hibernate.org/
[10] Talend data integration software, http://www.talend.com/

HOW TO MAINTAIN HUNDREDS OF COMPUTERS OFFERING DIFFERENT FUNCTIONALITIES WITH ONLY TWO SYSTEM ADMINISTRATORS

R. Krempaska, A. Bertrand, C. Higgs, R. Kapeller, H. Lutz, M. Provenzano
Paul Scherrer Institute, 5232 Villigen PSI, Switzerland.

Abstract

At the Paul Scherrer Institute, the control systems of our large research facilities are maintained by the Controls section. These facilities include the two proton accelerators, (HIPA and PROSCAN), the two electron accelerators, (SLS and the Injector Test Facility of the future SwissFEL) as well as the control systems of all their related beamlines and test facilities.

This paper describes methods and tools which are used to develop and maintain the challenging computing infrastructure deployed by the Controls section.

CONTROLS COMPUTING INFRASTRUCTURE

The Control system is basically composed of Input Output Controllers (IOCs) running VxWorks operating system on VME hardware using the EPICS (Experimental Physics and Industrial Control System) software. Additionally, there is an increasing number of non VME based IOCs, (so called EPICS soft-IOCs) running mainly on Linux hosts. In total the we are responsible for the installation, configuration and maintenance of up to 400 VME-based IOCs and more than 200 soft-IOCs. There is a remote console interaction with these systems in order to support their development, booting and debugging. Additionally, the control system includes special EPICS servers such as the CCD camera systems that often need dedicated configurations.

Finally, the client part of control system applications requires operator and expert consoles, login clusters and status displays. The Control system configuration and applications software for each facility is stored on independent NFS file servers. The total number of Linux computers and servers is about five hundred. Since only two system administrators are responsible for their installation, configuration and maintenance, we have adopted a well defined solution to face this challenging task:

- Virtualization
- Unified operating system installation and update mechanism
- Automatic configuration by a common tool (puppet)

VIRTUALIZATION

Virtualization allows the system administrators to configure a multitude of "one-task" simple machines such that administrative tasks can run on their own virtual computer. These virtual computers do not require a lot of CPU or memory. This is particularly useful for software upgrades and maintanence. Such systems include boot servers, soft-IOC servers, port server hosts and configuration servers (also called auto-save and restore servers). Computers providing access to private machine networks, such as secure shell (ssh) gateways and Channel Access gateways, are also installed as virtual machines.

Virtualization also enables a rationalization of hardware, operating and energy costs. Our virtual computers run on a VMware cluster (two servers, each with 72Gbytes RAM and SAN storage). Of the 500 Linux computers, about 200 are virtual machines running on the VMware cluster. The remaining systems, (NFS file servers, archiver or development servers) are installed on dedicated hardware blades.

UNIFIED OPERATING SYSTEM INSTALLATION

At PSI the popular Scientific Linux (SL) distribution is used [1]. The PSI Central Computing Division is in charge of distributing and maintaining the SL core and rpm packages [2]. SL is supported by various labs and universities around the world and is based on Red Hat Enterprise, which is recompiled from source and repackaged.

Using the so called "kickstart" mechanism of Red Hat Enterprise, we deploy the base operating system installation which is "tailored" by a software distribution mechanism, called puppet (see next section). The goal is to use the same system management framework for all hosts through their life cycle.

Configuration changes and software updates, (e.g. kernel upgrades) are steered from a central location. Operator consoles in the control room are installed on desktop computers with a standard installation in order to guarantee redundancy and a fast and easy exchange in the case of failure or upgrade.

Thus we are able to achieve an easy and uniform installation on all our systems.

AUTOMATIC CONFIGURATION BY PUPPET

Puppet is an automated administrative software engine for system configuration. It performs administrative tasks, such as adding users, installing packages, and updating server configurations, based on a centralized specification [3].

The Controls system administrators receive daily requests from users for new consoles, servers or office computers. Our existing 500 computers also need to be maintained. Without automated scripts, this would be almost impossible. The SL kickstart and subsequent puppet configuration is central to our automated installation process.

Each facility needs several types of computers, which we call classes. There are about 35 computer classes defined in puppet. The reason is that different classes of computer need different NFS mounts, environmental variables setup, services and cron tasks or additional local user administration and user accounts. A set of puppet configuration files stored in the central PSI repository handle these particular installations issues. Each computer configured by puppet is assigned a class and each class has a hierarchy of configuration files. An example of a configuration file for an IOC boot server configuration can be seen on Fig.1.

```
class class_A_BootPC_VM {

  info "PUPPETSERVERINFO: Configuring host $hostname, role $role, zone $gfaFacility:

  # GFA basics
  include "class_GFA_DesktopBasic"

  # Intranet AFS
  include "class_module_gfa_afs::intranet"
  include "class_module_gfa_files::mkhomedir_skel"
  include class_module_gfa_pam2::setup
  def_module_gfa_pam2::alternative{"GFA-local-noafs-krb5": activate => true, }

  include "class_module_gfa_runlevel::id3"
  include "class_module_gfa_iocLogServer"
  include "class_module_gfa_autoSR"

  # Accounts
  include "class_module_gfa_shellenv::sshkeys"
  include "class_module_gfa_account::ioc"

  include "class_module_gfa_vmware"

  include "class_module_gfa_epics::iocsh"

  # BootPC stuff
  include "class_module_gfa_bootpc"
  include "class_module_gfa_nfs::server"
  include "class_module_gfa_EpicsRecDB"
  include "class_module_gfa_dgrp"


  case $gfaFacility {
    SLS    : {
      def_module_gfa_bootpc::ioc_dir{"/ioc": }
    }
    FIN    : {
      def_module_gfa_bootpc::ioc_dir{"/ioc": mount => "/import_fin/ioc" }
    }
    HIPA   : {
      def_module_gfa_bootpc::ioc_dir{"/ioc": mount => "/import_hipa/ioc" }
    }
    Proscan : {
      def_module_gfa_bootpc::ioc_dir{"/ioc": mount => "/import_proscan/ioc" }
    }
    TRFCB : {
      def_module_gfa_bootpc::ioc_dir{"/ioc": mount => "/import_trfcb/ioc" }
    }
  }
}
```

Figure 1: An example of a puppet configuration file for a boot server virtual computer. The list of include commands in the first part shows the required configuration for this particular computer type. The case statement similar to the C code syntax in the second part shows how special treatment for each facility is done.

The configuration files are used when the puppet update process is scheduled from a cron-job (or manual request). Fig.2 illustrates a schematic flowchart diagram of the puppet deployment which can be factorized into the following steps:

1. The puppet client sends a request for an update. The "client identity", i.e. the computer name, or class is supplied.
2. The puppet server collects configuration information from the hierarchy of configuration files.
3. The puppet server sends packages and executes the update scripts defined in the configuration files.
4. The client gets the yum updates from the yum repository server.
5. The configuration results are stored in the Oracle database.

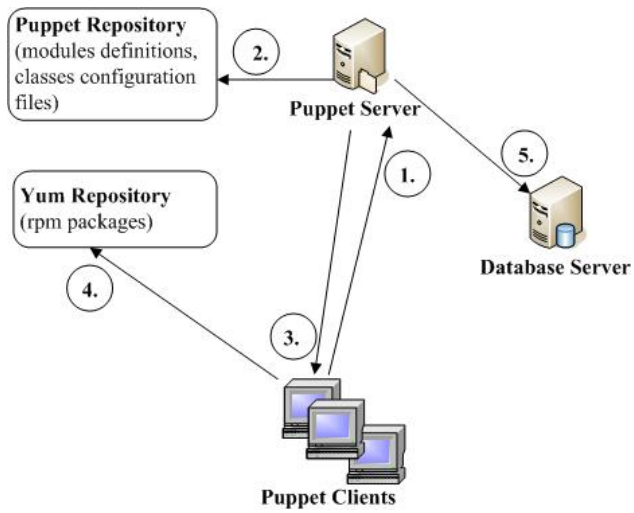


Figure 2: The puppet process configuration of the controls computers.

CONFIGURATION COMPUTERS OVERVIEW

The puppet configuration process is scheduled every 24 hours, typically during the night. Once the puppet configuration process is finished, the results are collected in the Oracle database. At the end of the process a database Web tool called Inventory software [4] integrates complete information about the configured hosts. This is a significant help for the system administrators. Fig.3 shows a database output list of all computers configured by puppet with supplementary information such as last update time, IP-address and Linux version. Finally Fig.4 displays even more detailed information for a host selected from Fig. 3.

Figure 3: List of controls computers configured by puppet.

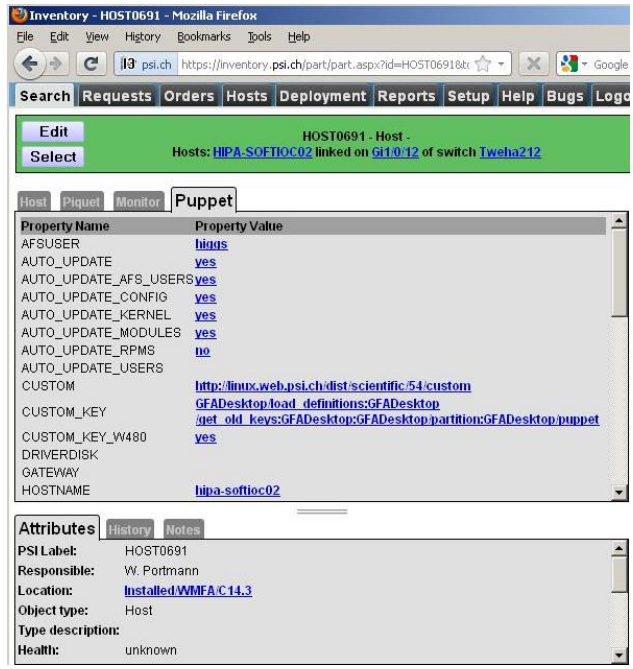


Figure 4: Detailed puppet configuration information for HIPA-SOFTIOC02 selected from Fig.3, visible on the PSI intranet Web.

ACKNOWLEDGMENTS

We want to acknowledge our colleagues M. Gasser, V. Markushin and H. Billich from the PSI AIT group, for their collaboration and for supporting the controls system administration.

REFERENCES

[1] http://www.scientificlinux.org/
[2] http://www.hpc-ch.org/wp/wp-content/uploads/2010/06/KS_Puppet_VM_20100520_printout.pdf
[3] http://projects.puppetlabs.com/
[4] http://gfa-it.web.psi.ch/invent_help/

INTEGRATED MANAGEMENT TOOL FOR CONTROLS SOFTWARE PROBLEMS, REQUESTS AND PROJECT TASKING AT SLAC

D. Rogind, W. Allen, W. Colocho, G. DeContreras, J. Gordon, P. Pandey, H. Shoaee
SLAC National Accelerator Laboratory, Menlo Park, California, U.S.A.

Abstract

The Accelerator Directorate (AD) Instrumentation and Controls (ICD) Software (SW) Department at SLAC, with its service center model, continuously receives engineering requests to design, build and support controls for accelerator systems lab-wide. Each customer request can vary in complexity from a small software engineering change to a major enhancement. SLAC's Accelerator Improvement Projects (AIPs), along with DOE Construction projects, also contribute heavily to the work load. The various customer requests and projects, paired with the ongoing operational maintenance and problem reports, place a demand on the department that consistently exceeds the capacity of available resources. A centralized repository - comprised of all requests, project tasks, and problems - available to physicists, operators, managers, and engineers alike, is essential to capture, communicate, prioritize, assign, schedule, track, and finally, commission all work components. The Software Department has recently integrated request / project tasking into SLAC's custom online problem tracking "Comprehensive Accelerator Tool for Enhancing Reliability" (CATER) tool. This paper discusses the newly implemented software request management tool – the workload it helps to track, its structure, features, reports, work-flow and its many usages.

CONTROLS SOFTWARE WORKLOAD

The ICD Software Department, comprised of about 25 engineers, provides complete, long-term, robust, software engineering solutions for control of distributed accelerator systems lab-wide. The department responsibilities include host computing infrastructure, networking, relational databases, real-time embedded (and host-level) device, subsystem, and instrumentation control, process control, engineering/operational GUIs, and service oriented architectures. It also performs user support, upgrade and maintenance activities. The Software Department uses the EPICS (Experimental Physics Industrial Control System) toolkit and its extensions as the basis for most software engineering implementations. The department also fully supports the legacy SLAC Linear Collider (SLC) VMS-based Control system.

ICD Software Department personnel currently spend, on average, 25% of their time in support of LCLS and FACET program operations. All SW (along with other Hardware (HW)) Problems (or trouble reports) newly entered into SLAC's CATER system are discussed at the daily 08:15 maintenance meeting and are usually assigned to a resource by a Lead within hours of creation. One software engineer per week, on a rotational basis, acts as a Controls Deputy (CD) liasion between LCLS Operations and the Software Department. The CD attends the daily meetings, work planning meetings, works and resides within the Control Room, and carefully helps to coordinate the weekly software releases.

For the remaining 75% of the time, software engineers perform development work which can be categorized as either a) customer requests or b) project work. Project work is typically funded and approved by the AIP or DOE (such as the LCLS II Construction Project). Until recently, customer requests were typically delivered verbally or via email to engineers; oftentimes the supervisor or lead remained unaware. Task scheduling and tracking then varied by individual or lead; commonly by a combination of memory, spreadsheets, lists, emails, and disparate project schedules. Both customers and supervisors had limited visibility into work planned, in progress, or accomplished. It remained difficult to assess whether additional resources were justifiable. Furthermore, the work was hard to prioritize and load balance without a centralized list of all department problems, requests, and project tasking

PROBLEM TRACKING TOOL

The CATER Problem Tracking Tool in its current implementation has been in wide use across many organizations at SLAC since 2008. The reports and workflow drive all of the daily LCLS (now joined by FACET) Operations/Maintenance meetings, software and hardware problem and maintenance work tracking, and contribute to the planning of the weekly machine development/access days. CATER is Oracle based, developed with Application Express (APEX), and has one full time developer (outside of ICD) allocated due to ever increasing feature demands.

In order to take advantage of CATER's existing user base, accounts, roles, relational database management, familiar work flow and familiar GUI, SW Requests (and their associated Tasks) were integrated alongside Problems into CATER in June, 2011. SW Requests are of two types: a) Customer, and b) Project Component. Additionally, a SW Job form was implemented which contains all fields required for software release approval and planning. This SW Job feature has proven very valuable for all software release types. Now for the first time, every software engineer has the capability of viewing all of their work – problems, requests, jobs, and project tasking in one place, via the CATER tool. Each user, upon logon, has a personalized dashboard on their home tab which displays all open assignments (Figure 5)

SOFTWARE REQUESTS AND TASKS

LCLS and FACET customers at SLAC are encouraged to login to the familiar CATER trouble reporting system and create a "New Software Request". There is a constant high volume of customer requests that are small in scale, of limited complexity and involve several people tasked separately. The automated Request system has been particulary efficient for tracking these types of entries that might otherwise get forgotten or not receive the proper visibility. Engineers/Leads also create entries to capture customer requests and components of their project based work. Requests use the existing CATER Problems schema and workflow. Fields include: *Title, Description, Status, Request Type ("Customer" or "Project Component"), Work Breakdown Description (funding source), Subsystem, Request Lead, Group (LCLS, FACET,..), Customer Priority, Customer Need Date, Created by, Modified by, Created Date, Modified Date.* There are buttons to *Add Customer*, *Add SW Job*, *Add Task*, and to *Upload* files.

A large Request, such as "Complete LCLS Fast Feedback Project" (Figure 1) may be further broken down into several Tasks, such as "Bunch Charge Feedback" (Figure 2), "Undulator Launch Feedback" etc. Each Task associates with a single resource and more detailed loading and scheduling information. Task fields include: *Title, Assigned to, Effort, Task Priority, Task Skill Set, Start Date, End Date, % Complete, Description, Status, Created by, Created Date*. There is enough information contained in the request /task database such that weekly reports and/or Gantt charts per individual or per project could be generated (future effort).

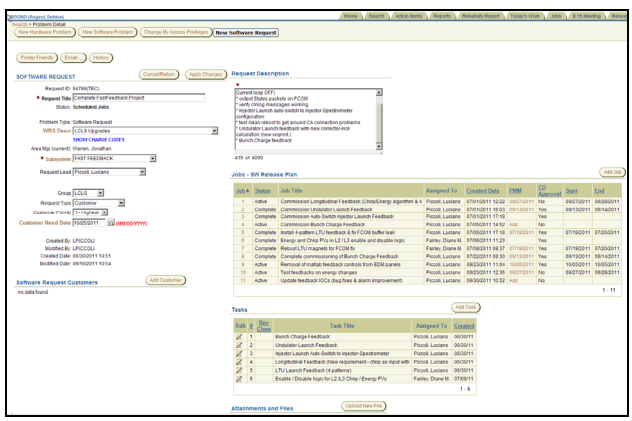


Figure 1: Sample Request Page.

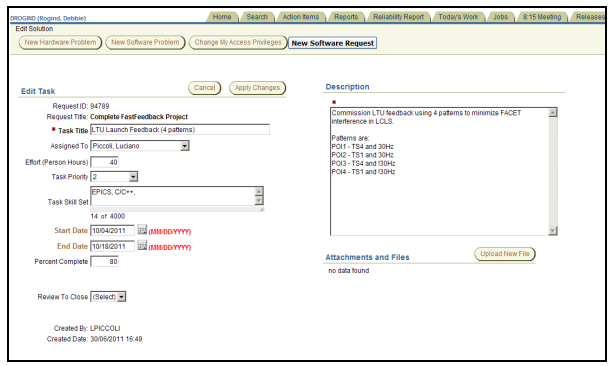


Figure 2: Sample Task Page.

SOFTWARE JOBS

LCLS hardware maintenance / repairs and software releases which require machine access (no beam) are typically scheduled every other week during user runs, while software releases / physicist machine development opportunities requiring an operational machine are scheduled once per week. To help maintain very high LCLS machine availability, all software releases are carefully evaluated for risk / benefit and are detailed in a new SW Job form that includes fields: *Job title, Resource, Time needed, Time Comment, Planned Start/Stop Date/Time, Access Requirements, Beam Requirements, Beam Comment, Invasive (Y/N), Invasive comment, Area, Subsystem, Test Plan, Back out Plan, Systems Required, Systems Affected, Risk/Benefit*. Refer to Figure 3 for a sample SW Job. The SW Jobs submitted each week are reviewed and scheduled in a weekly meeting between the program physicist and Controls Deputy; there are typically 15-20 SW Jobs of varying complexity submitted per week for LCLS. The Controls Deputy plays a pivotal role, especially on release day(s), coordinating the various software engineers along with parallel operational and physicist activities. (Refer to Figure 4, Sample Controls Deputy Report.)

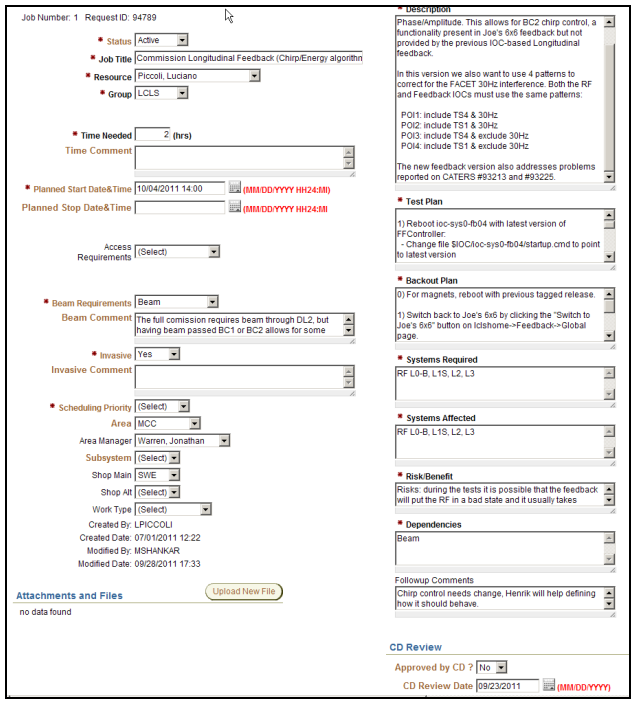


Figure 3: Sample Software Job (Release Plan) Page.

REPORTS

Once logged onto CATER, all users default to their Home tab (dashboard) that summarizes and links to all assigned open "Problems Assigned to Me", "Requests and Tasks Assigned to Me", and "Jobs Assigned to Me" (Figure 5). There are additional views associated with other tabs. The Search tab provides basic and advanced searches plus an interactive report builder to customize report column data and filters, such as All Customer Requests (224 in number) or All Project Requests. The Report tab has many canned reports including the Controls Deputy Report of SW Jobs up for weekly approval (Figure 4), Work Breakdown Structure Names/Priorities, etc. All reports export to MS Excel.

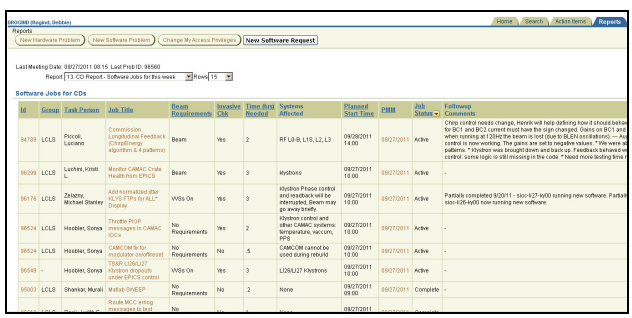


Figure 4: Sample Controls Deputy Report.

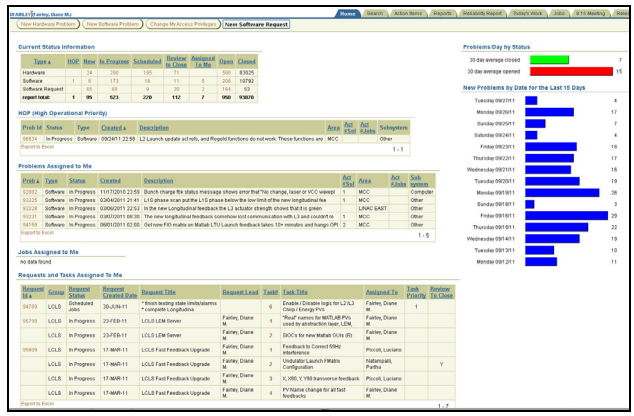


Figure 5: Sample CATER Home Tab (Dashboard).

CATER WORK FLOW

Requests follow the same simple, four state, CATER workflow as Problems because a) it has been proven to work effectively, and b) users are familiar with it. All status transitions in the life of a Request/Problem trigger alerts to email distribution lists which always include software leads. All New Requests/Problems have status=New. Once assigned to a resource, status changes to "In Progress" (or "Scheduled Jobs when SW Jobs are active). When all associated SW Jobs and Tasks (/Solutions for Problems) are complete, the status changes to "Review to Close". The Request/Problem is "Closed" by Leads with the appropriate closure privilege after careful review. The states, associated stakeholder review meetings, necessary reports, and email alerts during the life cycle of a Request/Problem are shown in Figure 6.

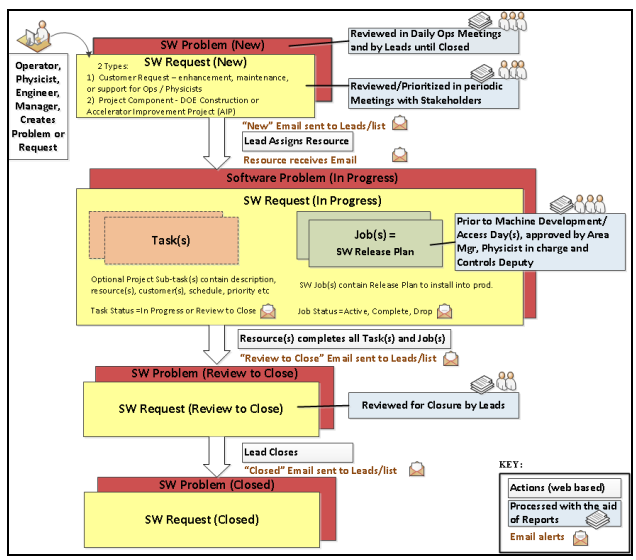


Figure 6: CATER Request/Problem Workflow.

CONCLUSION

Results Achieved

- Customers may view **all** department work to a) direct prioritization; b) better understand why their requests may not have resources allocated, and c) track and aid progress.
- Supervisors/Leads/Staff are informed of all ongoing / upcoming work. Helps to catch potential issues early, prioritize workload, re-task staff, track status, and justify resource decisions

Lessons Learned

- The tool has been particularly useful for tracking the large volume of requests that are small in nature and/or of minor complexity, and require a few hours to a few days of many different resources.
- Integration of Jobs (software test/release plans) for Requests and Problems has been very efficient and well-received for planning and executing machine development days
- Simultaneous development of CATER using Oracle's Application Express (APEX) proves challenging
- Releases for any new Request features are constrained by the normal CATER release cycle

What's next?

- More training and enforcement of data entry
- Roll out to other organizations within ICD; SLAC
- More reporting – weekly reports, Gantt charting

EXTENDING ALARM HANDLING IN TANGO

S. Rubio-Manrique, F. Becheri, D.Fernandez-Carreiras, J.Klora, L.Krause*, A.Milán Otero, Z.Reszela, P.Skorek, CELLS-ALBA Synchrotron, Cerdanyola del Vallès, Spain

Abstract

This paper describes the alarm system developed at Alba Synchrotron, built on Tango Control System. It describes the tools used for configuration and visualization, its integration in user interfaces and its approach to alarm specification; either assigning discrete Alarm/Warning levels or allowing versatile logic rules in Python. This paper also covers the life cycle of the alarm (triggering, logging, notification, explanation and acknowledge) and the automatic control actions that can be triggered by the alarms.

INTRODUCTION

ALBA is the first Synchrotron Light Source built in Spain [1]. Its 3 GeV Storage Ring has been commissioned during 2011 and it's expecting the first beam-line users for beginning 2012. Control of ALBA [2] is based on Tango, a modern control system for scientific facilities developed by the ESRF and maintained by the members of the Tango Collaboration [3]. Tango provides a collection of interfaces for common devices and generic tools for configuration, deployment, archiving and alarms.

The installation and commissioning of ALBA required alarm handling to supervise accelerators conditioning and detect changes in operation conditions. Although two alarm logging systems already existed in Tango, at ALBA we decided to have an alarm system integrated in the existing applications instead of having a separated tool. To do so we combined previous ideas and developments to provide a more flexible alarm handler.

ALARM SYSTEMS IN TANGO

The Tango Alarm System

The Tango Alarm System [4] was developed at Elettra institute (Italy) by Graziano Scalamera and Lorenzo Pivetta. It uses a MySQL alarm database containing sets of rules that are permanently checked by a central daemon, the Tango Alarm Server, which logs alarm changes and triggers actions if needed. The rules are combinations of boolean operators and Tango Attribute values.

Table 1: Tango Alarms Syntax

sr/psch/s1.1/highthr
(({sr/psch/s1.1/stat} & 0x80) && ({sr/psch/s1.1/curr} > 15.0))

*On leave

The Alarm daemon is connected with the control system using CORBA Notification events [4] triggered by the targeted Attribute device; avoiding overhead in the system. The Alarm Server provides logging and is capable to launch commands of other Tango Devices, in which notifications and actions are delegated. All alarms in the system are stored and can be visualized using the graphical log-viewer tool.

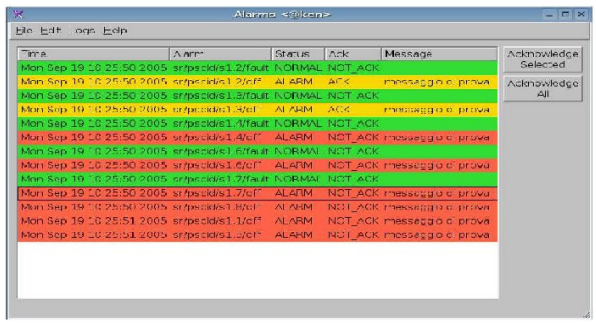


Figure 1: Tango Alarms Logs Viewer.

Soleil Alarm Database

The Alarm Database in operation at Soleil mimics the behaviour of Tango Archiving System [5][6], but it's focused on storing Attribute qualities (Valid, Invalid, Changing, Warning or Alarm) instead of values. The system uses a MySQL database and a pool of Archiver devices that poll periodically the quality of those Tango Attributes previously registered.

The conditions that trigger a change in Attribute Quality are not stored in the Alarm Database but in the Tango Database, using the Alarm/Warning Max/Min Values of each attribute. Configuration and visualization can be done using standard Tango Java tools and Archiving viewer.

ALBA PyAlarm

The PyAlarm device server development started on 2007 during ALBA's construction phase. It was focused on having an small stand-alone alarm system for installation activities (equipment tests, prototype labs, bakeouts, …) in places were database servers or network infrastructure was not fully available or could be interrupted.

Every PyAlarm Tango Device stores its rule sets as device properties, that can be stored in the Tango Database or Tango property files. These files allow the server to be running without Tango database if needed. Access between PyAlarm instances, clients and database have been encapsulated in the Panic API.

Alarm logging is done using Soleil Snapshoting database from the Tango Archiving System[5]. For every

Alarm triggered an Snapshot is recorded containing the Alarm Status and the value of every Attribute involved in the Alarm.

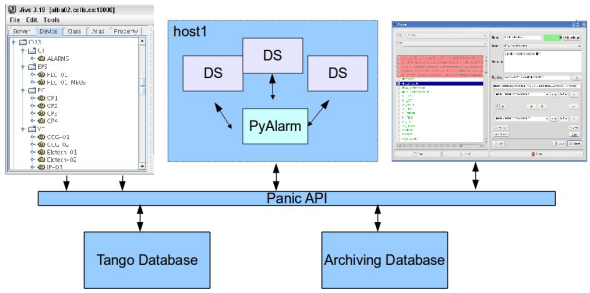


Figure 2: Panic API encapsulates database access and provides alarm setup, validation and visualization.

A Python Alarm System

The PyAlarm rules are inspired in Elettra's rule-sets, with the aim of integrating both systems in the future. But PyAlarm applies python parsing; enabling a richer rule syntax with list comprehensions, regular expressions, string replacement and other functional features.

PyAlarm also creates dynamically [7] new boolean attributes for each new rule set, used to display alarm states from any generic Tango client. The attribute names syntax has been extended to combine conditions on value, quality and time-stamp of attributes.

PyAlarm uses polling when running on top of PyTango, but if Taurus[8] library is available the PyAlarm can use it to switch transparently between polling or events for each attribute.

Table 2: PyAlarm declarations showing syntax for value, host, state, quality and regular expressions.

```
BL_PRESSURE:
                        BL/VC/VGCT-01/P1 > 3e-5
BL_LOST:
              tbl01:10000/BL/CT/DB/State==UNKNOWN
BL_TEMPERATURE:
              BL/EPS/PLC/T1.quality == ATTR_ALARM
ID_TEMPERATURE:
           any(t>85 for t in FIND(ID/*/*/Temp*))
```

THE ALBA ALARM SYSTEM

Although every PyAlarm is an independent process that runs stand-alone, all the Alarm system is coordinated using the Panic python API. This software layer encapsulates the access between servers, clients and the Tango database. The API provides a way to access alarms configuration and modify existing alarm distribution; not allowing to have duplicated alarms in the system.

Once configured, alarm logging and notifications are managed independently by each PyAlarm device. Each PyAlarm device instance manages a collection of alarms distributed by domain/family. Archiving and configuration are centralized in our Tango and Archiving Databases.

Accelerators Alarms Architecture

In Alba Accelerators the PyAlarm instances are dedicated by subsystem (e.g. being Magnets and Vacuum alarm systems are independent processes) and this instances can be decentralized deploying every PyAlarm in the same industrial PC were the monitored system is running.

Alarms can be declared hierarchically to summarize a big amount of alarms in fewer notifications. Alarms can be used as variables within other alarms formulas, that will summarize the state of their primitives when generating reports.

Beam-lines Alarms Architecture

An independent alarm system is running on each of the beam-lines. As the number of devices and subsystems is much smaller it is not needed to distribute the alarm system and it is centralized in a few PyAlarm servers running in the same virtual server were the Tango Database is running.

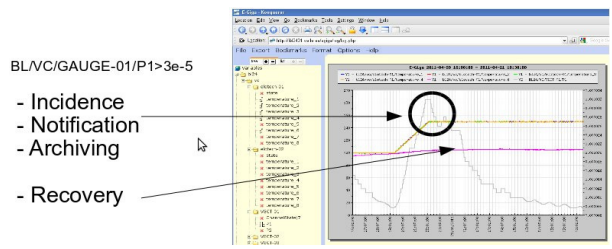


Figure 3: Messages sent during alarm life cycle.

The Alarm life-cycle

Alarms become active when the alarm condition evaluates to a True value, and will require human acknowledge to became inactive again. We added Reminder/Recovered notifications to avoid active and unacknowledged alarms remain unnoticed. Changes in alarm condition value will still trigger email and logging even if the alarm was still active, to make sure that no incidences remain hidden.

Table 3: Types of Messages

Alarm	The alarm condition has been activated.
Recovered	Alarm conditions are now inactive, but alarm state is kept.
Reminder	Condition was active for X period or became active after a Recovered period.
Acknowledge	Alarm has been acknowledged by operator.
Auto-reset	Alarm reset after being in Recovered state for a long time (optional).

Alarm Receivers

PyAlarm started as a notification service, so its main feature is email sending to notify any control incidence. It was initially extended to SMS and soon it was clear that more functionality should be needed. Every new notification feature has been added as a receiver type. In this sense an email address or a tango command are just different kind of receivers, that must be notified in case of incidence.

Every alarm has its own list of receivers, which may contain individual items or tags for groups of receivers. These groups are defined using email addresses, SMS numbers, lists of commands, archiving configurations or groups of them.

Table 4: Types of Receivers

email	Sent for every change in alarm status.
SMS	Sent only for activation.
Html files	To be loaded in the website.
SNAP	Record attribute values in the snapshoting database.
log	Generates a raw log file
COMM	Executes a TangoCommand.

USER INTERFACE

An easier method of global configuration/visualization of alarms helps accelerators and beamlines operators to diagnose incidences during commissioning; as well as modifying alarm conditions if needed.

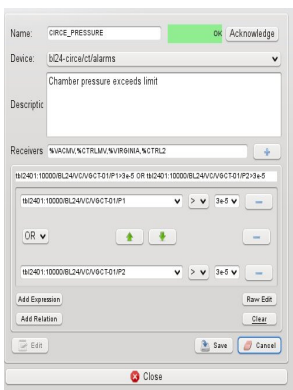


Figure 4: The Alarm editor widget.

An accelerator's alarm system requires an application for configuration, visualization and filtering of alarms usable at operator level; with no need of control system internals background.

The Alarm application allows to filter alarms by subsystem. To integrate alarms in existing applications has been developed an Alarm Toolbar with access to visualization and acknowledgement of alarms. This toolbar is able to filter alarms and status depending on user/application scope.

Alarms can be filtered by subsystem, receivers, attributes targeted and severity. Error, Warning, Info and Debug are the four severities available; which are used to sort the information when summarized in lists or toolbar. To go into detail as much as possible the alarm list provides viewer of attribute values.

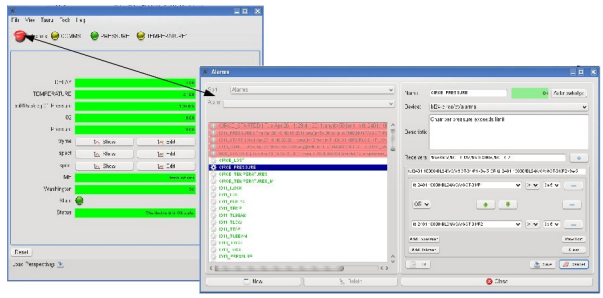


Figure 5: Alarm toolbar and list with filters and editor.

CONCLUSSIONS

We presented the ALBA Alarm System, created for ALBA installation and successfully extended to cover our accelerators and beam-lines. The PyAlarm was used successfully during installation and commissioning of ALBA linac, booster and beam-lines and in certain projects in the ESRF. The early deployment of an alarm system helped to prevent problems and detect irregular behaviours.

But for using it in our storage ring we had to improve the management of hundreds of alarms, adapting the content of messages and adding hierarchies between alarms that reduced the number of notifications sent. Our next objectives are the execution or recommendation of simple actions, linking alarms and low-level troubleshooting.

Unifying Tango alarm systems in a unique solution is still the aim of our development, so we focus next steps on interaction with existing systems; using PyAlarm as notification tool or adapting it to use Elettra Alarms Database as it exists now.

REFERENCES

[1] ALBA Paper D. Einfeld, “Progress of ALBA”, Proceedings of EPAC-2008, Genoa, Italy

[2] D.Fernández et al. “Alba, a Tango based Control System in Python”, ICALEPCS'09, Kobe, Japan.

[3] A.Götz, E.Taurel, J.L.Pons, P.Verdier, J.M.Chaize, J.Meyer, F.Poncet, G.Heunen, E.Götz, A.Buteau, N.Leclercq, M.Ounsy, “TANGO a CORBA based Control System”, ICALEPCS'03, Gyeongju, Korea

[4] Lorenzo Pivetta, “Development of the Tango Alarm System”, ICALEPCS 2005, Geneva, Switzerland

[5] E.Taurel, “Testing the Tango Archiving System”, Tango Meeting, 2004, ESRF, Grenoble, France

[6] S.Rubio et al, “Validation of a MySQL based archiving system for Alba Synchrotron”, ICALEPCS'09, Kobe, Japan

[7] S.Rubio et al., “Dynamic Attributes and other functional flexibilities of PyTango”, ICALEPCS 2009, Kobe, Japan

NFC LIKE WIRELESS TECHNOLOGY FOR MONITORING PURPOSES IN SCIENTIFIC/INDUSTRIAL FACILITIES*

I. Badillo#, M. Eguiraun, ESS-Bilbao, Spain
J. Jugo, University of the Basque Country, Spain

Abstract

Wireless technologies are becoming more and more used in large industrial and scientific facilities like particle accelerators for facilitating the monitoring and indeed sensing in these kind of large environments. Cabled equipment means little flexibility in placement and is very expensive in both money and effort whenever reorganization or new installation is needed. So, when cabling is not really needed for performance reasons wireless monitoring and control is a good option, due to the speed of implementation. There are several wireless flavors to choose, as Bluetooth, Zigbee, WiFi, etc. depending on the requirements of each specific application. In this work a wireless monitoring system for EPICS is presented. The desired control system variables are acquired over the network and published in a mobile device, allowing the operator to check process variables everywhere the signal spreads. In this approach, a Python based server will be continuously getting EPICS Process Variables via Channel Access protocol and sending them through a WiFi standard 802.11 network using ICE middleware. ICE is a toolkit oriented to build distributed applications. Finally, the mobile device will read the data and show it to the operator. The security of the communication can be improved by means of a weak wireless signal, following the same idea as in NFC, but for more large distances. With this approach, local monitoring and control applications, as for example a vacuum control system for several pumps, are currently implemented.

INTRODUCTION

Reliable, fast and secure communications must be assured in every place of a large scientific facility, even more when large amount of data is involved due to the rapid development of computing and electronics devices. Cables are often irreplaceable, but when they are not really needed for performance reasons, wireless is a suitable option for monitoring and control due to the numerous advantages it offers: flexibility, mobility, scalability, reduced costs and ease of maintenance.

Different wireless solutions can be found in the market. The most widely used band for industrial and scientific purposes is the ISM band. Inside this, ZigBee [1], Bluetooth [2] and WiFi can be found. The EEE 802.11 standard for WLAN, WiFi, is a very flexible technology, easy to implement, cheap and which provides a wide bandwidth. For these reasons, it has been implemented in large-scale systems, as presented in [3].

*Work funded by ESS Bilbao Consortium
#ibadillo@essbilbao.org

However, the radio waves used in wireless networks create a risk where the network can be hacked, making system vulnerable to threats as denial of service, spoofing or eavesdropping. These issues make mandatory the implementation of security mechanisms to minimize this drawback in industrial uses as SSL/TLS protocols and a proper architecture. In [4] an improved security mechanism is studied.

The goal of the present work is to build a secure and reliable human machine interface system for data monitoring purposes in a large scientific facility, based on a idea similar to Near Field Communication (NFC) but adapted to industrial needs. NFC offers a great security against external attacks since the signal makes physically inaccessible outside the range of transmission, so data exchange can only be made inside a limited radius.

The main disadvantage of this protocol for the present goal, is that the transmission distance, 4 cm or less [5], is insufficient for monitoring and control purposes.

In consequence, the proposed approach uses a limited field communication scheme, with WiFi technology and limiting the signal power to avoid external intrusions depending on the particular characteristics of each application.

In the presented schema, a distributed environment based on a TCP/IP network is considered, where an Experimental and Industrial Control System (EPICS) control network is implemented, [6]. The system sends the desired data over the WiFi network and publishes it in an Android based mobile device, which must be located inside the wireless physical transmission range. Two-way communication will allow not only monitoring, but also changing signal values, for example to turn on/of a certain device.

The idea has been implemented for monitoring the vacuum control system of a negative ion source.

PROPOSED APPROACH

When designing a wireless communication system, security becomes critical, even more when talking about large scientific facilities. In these environments, such as ion sources and particle accelerators, intrusions may result in harmful or even disastrous situations due to the large amount of power involved in equipment consumption. In order to avoid undesired failures or data losses, developers of wireless standards incorporate a large variety of security related features in the protocols. Two of the most commonly used tools is message encryption and node verification. This is used in order to maintain

data integrity, prevent interception of the transmitted data between nodes of the network and avoid spoofing.

Another way to provide security to a wireless network, related to its architecture, is to adjust the transmit-power level to control signal spillage beyond the plant walls. If the radio signal is "invisible" beyond the limits of the facility, it becomes very difficult to steal or intercept the signal. That means a physical security against attacks, independently from the software security mechanism that will be implemented.

This idea is represented in Figure 1, where the mobile device located outside the transmission range cannot reach the wireless signal, therefore it is impossible to access information. It also allows to spread signals only in certain areas of the facility depending on the authorization level. So, a limited field communication approach can lead to a secure installation, limiting the transmission power accordingly to the characteristics of each area.

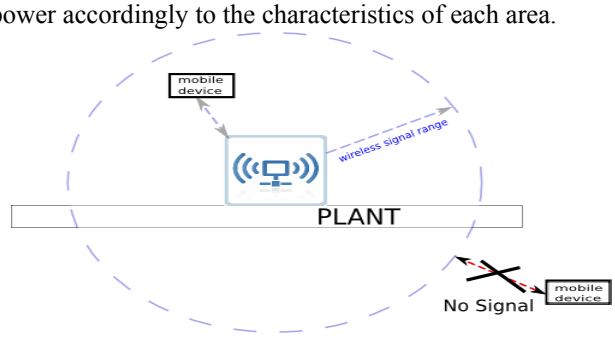


Figure 1: Security based on adjusting transmit-power level.

PROPOSED APPROACH IMPLEMENTATION

Large scientific facilities are complex distributed systems where important amounts of data must be processed and different control devices must be integrated. In this context, as every element might have independent behavior, a middleware distributing messages, commands signals and/or requirements over the net and between elements can be very helpful.

Nowadays, a large number of scientific facilities are being built using EPICS as middleware layer. Its wide usage, makes this solution very scalable. In fact, many vendors incorporate EPICS drivers in their products. Moreover, if a custom device is needed, in house development is also possible. This is the main control system used in the present limited field communication application.

On the other hand, the reduction in production cost of electronic devices and new technologies during last years has open a new market for mobile devices, such as tablets. In a large facility, a small and light computing device, with resources for networking environment, can help the operator in a lot of ways in both the usual operation and maintenance tasks. In this sense, Android based tablet has been chosen as mobile monitoring device, since its popularity and availability of many IDE environments make easy the development of custom applications.

Finally, as no libraries of EPICS for Android are available (unlike for iOS[7]), the use of the ICE middleware [8] is needed to integrate such different environments. It provides libraries for many programming languages, but, in addition to this, network security issues are a key issue of its functionalities.

The main characteristics of these technologies are summarized in the following paragraphs:

- EPICS: It is a control solution based on middleware approach, oriented to distributed control systems. It is used worldwide to create soft real time control systems specially for large scientific facilities as particle accelerators and telescopes. EPICS can be defined as an architecture for building scalable control systems, a collection of tools and codes made collaboratively between major science labs and industry. It is free and reliable and it is being more and more chosen to implement control systems in strategical projects as ITER (International Thermonuclear Experimental Reactor) or ESS (European Spallation Source). That eases feedback between developers in order to improve it. As mentioned before, the proposed networked control system architecture is based on a TCP network due to its advantages in cost and easy integration. But this protocol has non-deterministic characteristics, which makes difficult its use in control systems. The use of EPICS minimizes these disadvantages. Several EPICS controllers (IOCs) are spread along the facility, associated to different devices: sensors, DAQ systems and so on. These IOCs communicate among themselves and share information (Process Variables or PVs) using a protocol called Channel Access (CA) over a TCP/IP standard network. There are several security related mechanisms in EPICS, from among which highlights the EPICS Gateway. It is both a CA server and client and allows to separate the EPICS network from the network it is accessing from.
- ICE (Internet Communications Engine): This is an object-oriented toolkit for building distributed applications, heir of CORBA. It allows to communicate two or more applications of very different nature (operative systems, programming languages...). ICE offers different solutions for security, as encrypted communications and authentication through SSL, which is supported in all of the ICE language bindings.
- Android: Android is a mobile operative system based upon a modified version of the Linux kernel. Since the computing power of the mobile devices is quickly increasing and the price is reducing, its usage is fast spreading in different fields as industrial and scientific facilities [9]. Android is one of the leading OS for this kind of devices and it is updating and

adding services day by day, offering to the developers a huge number of tools to create any types of applications. The monitoring application is developed on an Android platform.

Implemented Architecture

In the implemented schema, a Python based server is used as a "bridge" between EPICS and the Android mobile device. This server program acquires the EPICS Process Variables over the network using the EpicsCA library [10], which provides methods for reading/writing PVs from Python via CA protocol. Moreover, the program organizes the captured PVs in a proper structure to ensure a good throughput and initializes the ICE host application. It creates the ICE object that responds to the client requests.

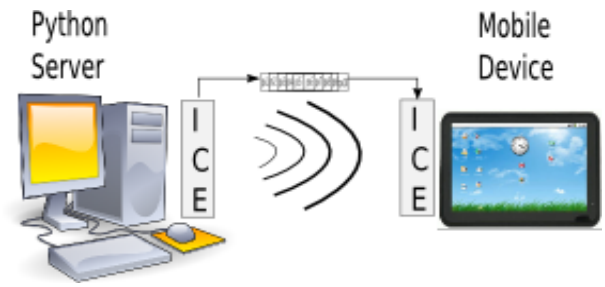


Figure 2: Client/Server communication using the ICE middleware.

Finally, the client is running in an Android mobile device. This application, written in Java, creates a proxy to connect to the object located in the server and invokes the needed operations to request the data. Once the data is obtained, the user interface displays the desired information to the operator. The Figure 2 shows a visual representation of the implemented schema.

SAMPLE APPLICATION: ISHN VACUUM SYSTEM MONITORING AND CONTROL

The presented application is intended to be used on a large scientific facility, to ease the task of the operators during normal operation and in maintenance stages. The main idea is to allow them to control and monitor the main variables of a vacuum system wherever they are inside the spread radius, depending on the transmission power. This fact allows to avoid the dependence of a central computer. Specifically, it has been designed to monitor the vacuum control system of the ISHN (Ion Source Hydrogen Negative) project at ESSBilbao, [11].

ISHN project consists of a Penning type ion source which will deliver up to 65mA of H− beam to a linear accelerator, which finishes generating neutrons by means of spallation process (currently under design). Apart from main devices for managing the ion source, for instance power supplies for plasma generation and hydrogen feed system, the vacuum system in considered a critical system of the project. For the ion source operation, a pressure value in the order of 10^{-6} mbar is required. Any failure could cause dramatic accident, due to the high voltage values needed for operation (several kilovolts), because a vacuum loss could cause high voltage breakdown.

In order to reach the desired vacuum level two mechanical and two turbo pumps are used. Their control system is isolated from the local control of the ion source, except for some interlock and emergency signals. There are managed by a PLC from Schneider, which reads all the signals involved in the operation of the pumps, such as rotation frequency, pressure level, status, etc. A Modbus TCP/IP server publishes all of these variables, and some of them can be accessed in both write and read mode.

An EPICS IOC accesses all of the data by Modbus TCP/IP calls [12], which acts as a wrapper for Twido communication. Even if Modbus communication is present, any call to vacuum system can only be done through EPICS, ensuring that capabilities offered by EPICS are always met.

In such configuration, the devices controlled with EPICS are two mechanical pumps, two turbo pumps, a vacuum sensor and two vacuum gates.

The application for Android has been developed in Eclipse using the ADT (Android Development Tools) and tested on the emulator provided on the Android SDK [13]. The program has been written for the API Level 12 (Android 3.1), compatible with all the newer version and APIs. All the libraries used in the project are provided with the ADT, except the ICE specific ones and those used to build the XY plots. These last were imported from the "androidplot" project [14], which offers a pure Java API for creating dynamic and static charts within Android applications.

The GUI shows a text box where the user must enter the IP to connect to and a "connect" button. Most of the variables are booleans as on/off, alarm status etc., so they will be displayed using a widget that emulates a led, as shown in Figure 3. The analog signals, as rotation frequency of the turbo pumps or temperatures, are represented as a waveform chart which refreshes itself dynamically.

CONCLUSION

In this work, the use of wireless communications for monitoring and sensing in large industrial and scientific facilities has been discussed, proposing a limited field communication approach. In addition, a particular application of this technology has been presented. The advantages of wireless devices make this technology an interesting alternative to common wired and fixed monitoring stations. However, as long as wireless communications are involved, security becomes critical. Adjusting transmission power allows to spread the signal only in the desired radius, avoiding external attacks. A good mechanism limiting the transmission power depending on the application characteristics is also necessary.

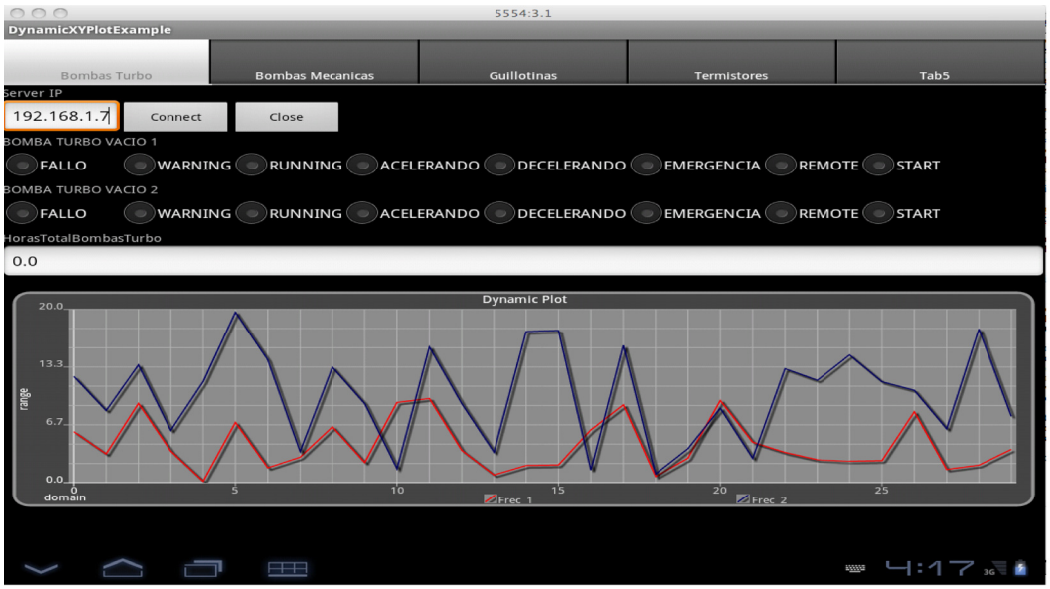


Figure 3: GUI of the Android application.

In particular, EPICS network distributed applications, which involve very heterogeneous environments, take advantage of the simplicity and capabilities of ICE toolkit and the versatility of Android based devices for LFC implementations.

The main purpose of the future work is to implement encrypted communications and authentication through SSL protocol. There is also projected to integrate the EPICS monitor system in a Python based server, to avoid polling in order to improve the overall throughput.

It is worth noting that the presented application will be implemented in the ESS Bilbao project for a real usage and it is intended to extend its monitoring and control tasks to several systems apart from the vacuum control system of ISHN.

REFERENCES

[1] W.-T. Sung and Y.-C. Hsu. Designing an industrial real-time measurement and monitoring system based on embedded system and ZigBee. EXPERT SYSTEMS WITH APPLICATIONS, 38(4):4522–4529, APR 2011.

[2] Bluetooth home page, http://www.bluetooth.com

[3] E. Witrant, A. D'Innocenzo, G. Sandou, F. Santucci, M. D. Di Benedetto, A. J. Isaksson, K. H. Johansson, S.-I. Niculescu, S. Olaru, E. Serra, S. Tennina, and U. Tiberi. Wireless ventilation control for large-scale systems: The mining industrial case. INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL JAN 25 2010.

[4] R. Falk and H. J. Holf. *Industrial sensor network security architecture*. In *Emerging Security Information Systems and Technologies* (SECURWARE), 2010 F*ourth International Conference* on pages 97-102, july 2010

[5] E. Haselsteiner and K. Breitfuß. Security in near field communication (nfc). RFID sec, 2006.

[6] EPICS home page, http://aps.anl.gov/epics.com

[7] capod, EPICS CA libraries for iOS, sourceforge.net/projects/capod/?_test=beta

[8] Zeroc ice middleware, http://zeroc.com

[9] A system architecture for industrial application based on the Android Mobile OS, A. Sbaa, R. El Bejjet, H. Medromi

[10] Pyepics home page, http://cars9.uchicago.edu/software/pyton/pyepics3

[11] Ishn web, http://essbilbao.org

[12] EPICS Modbus, Mark Rivers, http://cars9.uchicago.edu/software/epics/modbus.html

[13] Android, http://developer.android.com

[14] Androidplot project, http://androidplot.com

APERTURE METER FOR THE LARGE HADRON COLLIDER

G.J. Müller, K. Fuchsberger, S. Redaelli, CERN, Geneva, Switzerland

Abstract

The control of the high intensity beams of the CERN Large Hadron Collider (LHC) is particular challenging and requires a good modeling of the machine and monitoring of various machine parameters. During operation it is crucial to ensure a minimal distance between the beam edge and the aperture of sensitive equipment, e.g. the superconducting magnets, which in all cases must be in the shadow of the collimator's that protect the machine. Possible dangerous situations must be detected as soon as possible. In order to provide the operator with information about the current machine bottlenecks an aperture meter application was developed based on the LHC online modeling toolchain. The calculation of available free aperture takes into account the best available optics and aperture model as well as the relevant beam measurements. This paper describes the design and integration of this application into the control environment and presents results of the usage in daily operation and from validation measurements.

INTRODUCTION

The LHC is running with outstanding performance [1], currently being operated with small emittances and above nominal bunch population (up to 1.4×10^{11} p). Beams with a stored energy of 110 MJ are brought in collision and produce peak luminosities up to 3.3×10^{33} cm^{-1}s^{-1}. This allows to deliver integrated luminosities topping 110 pb^{-1} per fill. Providing very good conditions for the experiments to accumulate collision data, the high energy beams impose a severe danger for the accelerator (especially the superconducting magnets) which has to be protected against beam loss induced damage.

A complex machine protection system is put in place [2], which prevents damage to the accelerator equipment by extracting the beams as soon as any of the safety systems (beam loss monitors, magnet current monitors, quench protection system, collimation system,...) detects potentially unsafe conditions. The knowledge of the available clearance around the circulating beams is of primary importance for machine protection in case of problems related to orbit and optics changes that can bring the beams too close to sensitive equipment. To help the operation team to detect critical conditions early on, an Aperture Meter was conceived and developed to monitor the machine aperture online. After introducing the concept of machine aperture, the design of the LHC Aperture Meter is presented and some first results from standard LHC physics fills and from aperture measurements are shown.

APERTURE DEFINITIONS

The definition of the mechanical aperture of each lattice element seen by the beam is referred to as aperture model of the machine. A continuous model that covers the 27 km of each of the two rings (B1/B2) is available for the LHC.

The concept of available clearance for the circulating beams is however more complex and requires taking into account other relevant machine and beam parameters in addition to the detailed knowledge of the aperture model. The main ingredients to evaluate the aperture in the two planes $z = [x, y]$ at a location s are (see illustration in Fig. 1):

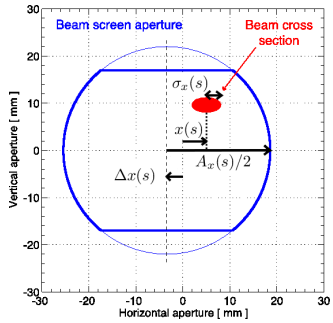


Figure 1: Beam screen aperture with relevant aperture definitions (horizontal plane)

a) mechanical aperture $A_z(s)$ from the design specification or calculated from the movable device positions;
b) offset $\Delta z(s)$ to take into account imperfections like misalignments of magnets (from cold bore measurements before installation) or to include the outcome of beam based alignments of the movable devices that will be described more in detail later on;
c) beam position $z(s)$ and transverse beam size $\sigma_z(s)$.

The minimal beam clearance or available aperture $a_z(s)$ is calculated per plane and defined as

$$a_z(s) = \frac{A_z(s)/2 - |z(s) - \Delta z(s)|}{\sigma_z(s)}$$

For convenience, the aperture is normalized to the transverse beam size $\sigma_z(s) = \sqrt{\beta_z(s)\epsilon_z^{design}}$, where we use the design emittance ϵ_z^{design} for the normalization. Thus the available aperture is expressed in terms of number of available design beam σ. The minimum normalized aperture $a_z^{min} = \min_s a_z(s)$ for each beam and plane is referred to as the aperture bottleneck, e.g. four bottlenecks exist for the LHC.

The method for calculation of the available aperture presented here is a simplified approach to estimate online the clearance for the beams. Information of the nominal machine (optics functions and the design aperture model) are merged with online measurements (movable device positions, beam position, beam energy and the state of the machine with respect to optics and beam trajectory configuration).

DATA PROCESSING

Operational Cycle

During the operation cycle [3] the LHC passes four static states of undefined length to reach physics conditions: i) *Injection* when beams are injected at 450 GeV, ii) *Flat-top* after the beams have been accelerated to the collision energy

of 3.5 TeV, iii) *Betatarget* when the optics have changed so that the β function at the interaction point (IP) is reduced from 11 m to currently 1.0 m β* and iv) *Physics* after the parallel separation at the IPs is collapsed to put the beams in collision. In these static states adjustments to the machine can be performed. Four states with pre-defined time duration exist as the transitions between them: a) the energy *Ramp* to *Flattop*, b) the *Squeeze* to *Betatarget* and c) *Collisions* to reach the *Physics* state. To perform the transition, setting functions with pre-defined length are loaded and executed in the hardware (e.g. power converters, collimator's,...). The `OnlineStateProvider`, provided by the online modeling toolchain (OMT) [4], is used in the aperture meter to determine the current state and time in the setting functions.

Beam Trajectory and Beam Size

The interpolation of beam position data measured by the beam position monitors (BPMs) to all machine elements is done by using the orbit interpolation with transfer matrices implemented in JMad, the JAVA API to MAD-X [4, 5]. A mechanism is provided that allows to define which monitors should be used for the interpolation. This feature is used to exclude BPM data that is flagged as bad reading by the data acquisition or data from BPMs that are excluded by the user. The `JMadOnlineService` of the OMT is used to provide the current machine state's optics functions to be used for the calculation of the transfer matrices. The current machine energy is provided in the service as well. Together with the available nominal β function it allows the calculation of the beam sizes.

Collimator's and Aperture Model

A sophisticated multi-staged collimation system is installed in the LHC for beam cleaning and to provide passive machine protection. Primary (TCP), secondary (TCS) and absorber collimator's (TCLA) are installed in the cleaning insertions of point 3 and 7. Tertiary (TCT) collimator's are installed in the IP regions to complete the collimator hierarchy and protect the triplet magnets. Special collimator's are installed in the injection and dump regions. The collimation hierarchy is defined such that TCPs are the closest to the beam, followed by the TCS, TCLA and TCT. This hierarchy ensures that beam cleaning losses are caught by collimator's only, protecting the superconducting magnets and other equipment. Dangerous conditions that are difficult to observe can develop as a combination of e.g. an orbit bump outside the collimator's and a beam impact at another location. These are not covered by the collimation system and would cause beam losses at the bump location sufficient to create damage – another reason for continuous aperture monitoring.

Collimator's generally have two jaws, controlled by very precise stepping motors, that are aligned around the beam during collimator alignment campaigns. Procedures are established [6] to determine the beam based collimator center. These alignments have to be performed at each static state in the operational cycle to produce the input for the collimator setting generation for the functional states of the operational cycle [7]. Based on these settings, the collimation system then follows the orbit (beam separation and crossing) and optics evolution in the operational cycle. The collimator jaws are aligned around the measured beam centers with the required jaw openings.

The online aperture model provided by the OMT [4] is an extension to the aperture model and calculates the mechanical aperture $A_z(s)$ from the measured collimator jaw positions. The discrepancy between the interpolated orbit in the collimator's during the alignment and the observed beam-based collimator center (the reference for the jaw movement) is taken into account with the offset $\Delta z(s)$. The following procedure is used to determine $\Delta z(s)$ for all collimator's and operational states: i) Establish a default monitor setting for the orbit interpolation module. ii) Interpolate the measured orbit at the time of the collimator alignment to the collimator positions. iii) Store the differences between interpolated orbit and measured beam based centers as $\Delta z(s)$ for the active state. For each alignment campaign performed at the static operational states, step ii) and iii) are repeated. During operation a mechanism is put in place that ensures that the correct offsets are loaded according to the current operational state. If a functional state is active, a linear interpolation to the current time in the setting functions is performed between the offsets of the two adjacent static state.

DESIGN AND FEATURES

The aperture meter application is split in a service and a GUI implementation. As the input data from the various sources is not synchronized, a `DataStore` is implemented in the service and filled independently with: beam sizes, simulated orbit and machine state from the `JMadOnlineService` (currently when triggered and changed data available), aperture data from the online aperture model (when movable device positions change) and measured beam positions (1 Hz) from the Orbit Feedback Service Unit. A `DataSynchronizer` is triggered every 3 s to collect all data from the `DataStore`, perform the orbit interpolation, calculate the available aperture at all elements and determine the five smallest values $a_{z,i}^{min}(s_i)$ of $a_z(s)$ per beam and plane in the machine. The whole data set is then published to all registered listeners like the GUI.

A logging mechanism is implemented, to save the data set to xml files. In playback mode the aperture meter can be used to load either complete aperture meter data sets or to drive the aperture meter with the information from logged orbit data (logging database or file). Additionally the JMad GUI can be started from the aperture meter to perform virtual trims on the knobs that have been imported by the `JMadOnlineService` as well as to change optics.

Various displays are provided in the GUI to visualize the aperture meter data. The main display, conceived as fixed display, is composed out of four minimal aperture evolution views like the one in Fig. 2, one for each beam/plane

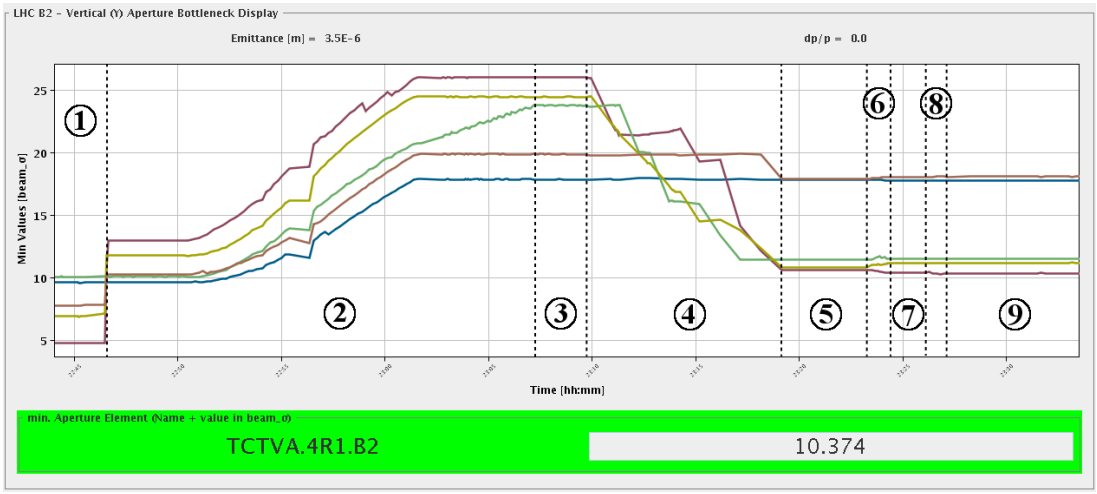


Figure 2: Minimal aperture evolution for beam 2 vertical plane over the operational cycle. The primary and secondary collimator positions are not taken into account.

combination. Each of these views shows the evolution of the available aperture $a_z(s)$ over time for the five locations (containing the bottleneck) in the machine having the smallest values $a_{z,i}^{min}(s_i,t)$ in the assigned beam/plane. Depending on the changes in the machine, the locations of $a_{z,i}^{min}(s_i,t)$ change. To broaden the provided information, only the element with the smallest value per assembly is selected during the minimal aperture determination.

MEASUREMENTS

Two examples for the visualization of the available aperture, calculated in the aperture meter, are presented in the following to demonstrate the functionality of the application.

Full Cycle of the LHC

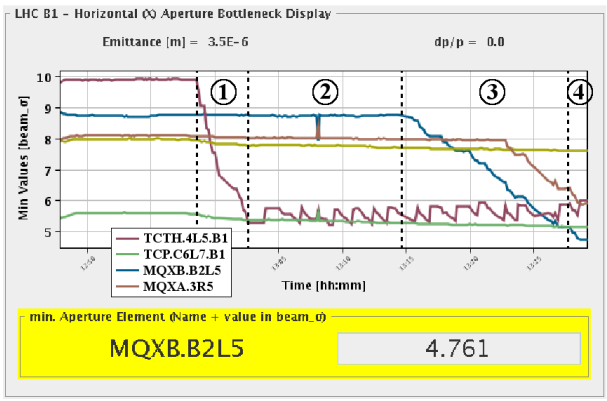
The evolution of $a_{z,i}^{min}(s_i,t)$, calculated by the aperture meter during the operation cycle of the LHC is shown in Fig. 2 for beam 2 in the vertical plane. To visualize the adjustments of the tertiary collimator's (TCT) during the operational cycle the primary (TCP) and secondary (TCS) collimator $a_z(s)$ information have been excluded from the selection of $a_{z,i}^{min}(s_i,t)$, they would be the dominant machine elements as they are the closest to the beam according to the collimator hierarchy. The cycle shown in Fig. 2 brings the beams into collision with no squeeze in IP2, squeeze to 3m in IP8 and to 1.5m in IP1 and IP5. The numbered segments represent the following operational steps: (1) Beam *Injection*, finalized by the retraction of the injection protection devices which can be observed by the step change at the end of the segment. Absorber collimator's (TCLA) in point 3 and 7 and the TCT collimator's in the vertical plane are the $a_{z,i}^{min}(s_i,t)$ locations afterwards. (2) Energy *Ramp* to 3.5 TeV. Caused by *adiabatic damping* the beam size shrinks with increasing beam energy, observable by the increasing $a_{z,i}^{min}(s_i,t)$. The sharp edges might be caused by orbit changes, but have to be investigated further. (3) *Flat-top*, const. $a_{z,i}^{min}(s_i,t)$ and prepare *Squeeze*. (4) *Squeeze*, the β function changes around the IPs and the TCTs are moved closer to the beam. (5) *Betatarget*, const. $a_{z,i}^{min}(s_i,t)$, prepare *Collisions*. (6) *Collisions*, parallel separation closed which can be observed by small changes in $a_{z,i}^{min}(s_i,t)$. In segments 7, 8, and 9 physics conditions are reached. Segment 8 shows marginal changes of $a_{z,i}^{min}(s_i,t)$ caused by luminosity scans. The transitions in the *Squeeze* are not smooth as the adjustment to changed optics takes $\approx 45s$.

Triplet Aperture Measurements

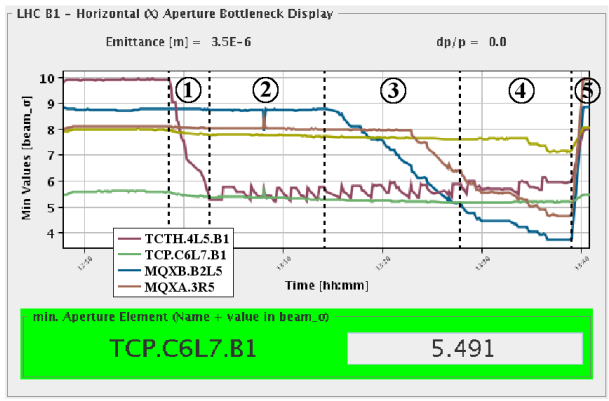
To check and verify the triplet aperture before operating at reduced β* (the extremal β-function in the interaction points) of 1.0 m, scans have been performed in the triplets by using the angle and separation scan knobs created for the luminosity optimization and leveling. These knobs are defined for both beams and have been trimmed simultaneously to create bump shapes comparable to the nominal crossing and separation. The following steps were performed during the scan:

a) increase the bump amplitude until one of the beams touches the corresponding tertiary collimator (TCT), observable by the created beam losses;
b) retract both TCTs left and right from the IR by 0.5 σ;
c) repeat steps (a),(b) until the triplet aperture is exposed, observable by larger loss spikes in the triplet magnets.

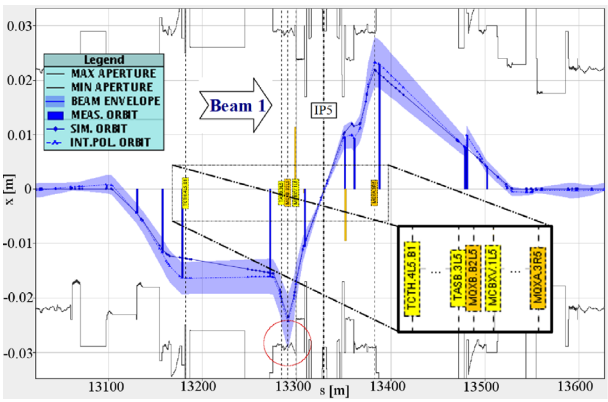
This procedure can be nicely observed by the aperture meter as shown in Fig. 3. The Figures 3a and 3b show the evolution of $a_{z,i}^{min}(s_i,t)$ during the scan. Figure 3a is an example how the bottleneck display notifies about $a_{z,i}^{min}(s_i,t)$ below the warning limits (currently set to 5.0 beam σ). The evolution plots show that under normal conditions the primary collimator (TCP.C6L7.B1) is the bottleneck a_x^{min} in the horizontal plane for beam 1. When the scan is started (1) the TCT is moved towards the beam until it becomes a_x^{min}. The approach of the aperture with the scan according to step (c) can be observed in (2) without any impact on the $a_{z,i}^{min}(s_i,t)$ except for the moving TCT. In (3) the two magnets of the triplets right and left of the IP appear in



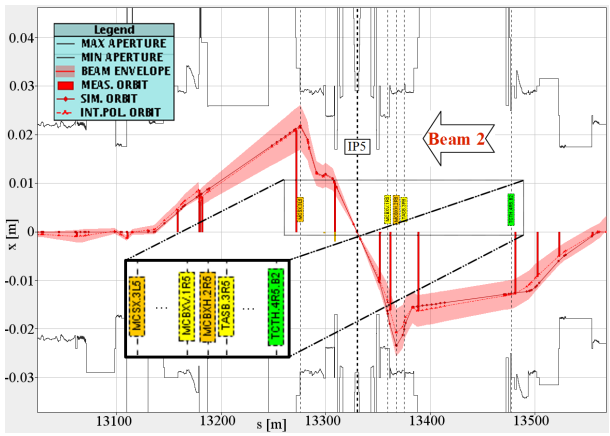
(a) Minimal Aperture Evolution: Triplet Exposed.



(b) Minimal Aperture Evolution: Full Aperture Scan.



(c) Beam 1 display: Max. Hor. Bump.



(d) Beam 2 display: Max. Hor. Bump.

Figure 3: Horizontal triplet aperture scan in IP5 (CMS) at 1.5m β*

$a_{z,i}^{min}(s_i,t)$. The magnet MQXB.B2L5 of the triplet right from the IP becomes a_x^{min} in (4) and the beam is gradually moved closer, reducing a_x^{min}. Finally the triplet is exposed, observed by the created losses. In (5) the collimator's are moved out and the scan knob is trimmed back to zero, to remove the orbit bump and recover nominal operational conditions.

Figures 3c and 3d are the beam views of the aperture meter at the maximum bump excursion achieved during the horizontal scan performed in IP5 (CMS). The simulated, measured and interpolated orbit is plotted in the visualization of the mechanical aperture model. A 3 σ beam envelope is added to the interpolated orbit. The result in the aperture meter (beam envelope touches aperture of MQXB.B2L5 – red circle in Fig. 3c) agrees with the beam loss observations during the scan.

CONCLUSION AND OUTLOOK

The principles for minimal aperture calculation and the design of the aperture meter application have been presented. A first implementation is available for operation and has been especially useful for aperture measurements. Validation is ongoing to determine the accuracy of the calculation results. Further work has to be devoted to optimize the configuration of the system and to improve the visualization and playback features.

ACKNOWLEDGMENTS

The authors thank M. Giovannozzi, F. Schmidt, M. Strzelczyk, R. Tomas and J. Wenninger for fruitful discussions and useful advice as well as the collimation team for the provision of the collimator alignment data. This work was supported by the Wolfgang-Gentner-Programme of the Bundesministerium für Bildung und Forschung (BMBF).

REFERENCES

[1] M. Lamont, *The LHC from Commissioning to Operation*, Proceedings of IPAC'11, 2011.

[2] R. Schmidt *et al.*, *LHC Machine Protection*, Proceedings of PAC'07, 2007.

[3] S. Redaelli, *How to improve the turn-around.*, Proceedings of 2nd Evian Workshop on LHC Beam Operation, 2010.

[4] G. Müller, *et al.*, *Toolchain for online modeling of the LHC*, these proceedings.

[5] K. Fuchsberger *et al.*, *Status of JMad, the* JAVA-*API for* MAD-X, Proceedings of IPAC'11, 2011.

[6] D. Wollmann *et al.*, *First Cleaning with the LHC Collimators*, Proceedings of IPAC'10, 2010.

[7] R. Bruce *et al.*, *Principles for Generation of time-dependent Collimator Settings during the LHC cycle*, Proceedings of IPAC'11, 2011.

STABILIZATION AND POSITIONING OF CLIC QUADRUPOLE MAGNETS WITH SUB-NANOMETRE RESOLUTION*

Stef Janssens†, Kurt Artoos, Christophe Collette, Marco Esposito, Pablo Fernandez Carmona, Michael Guinchard, Claude Hauviller, Andrey Kuzmin, Raphael Leuxe, Raul Moron Ballester, CERN, Geneva, Switzerland

Abstract

To reach the required luminosity at the CLIC interaction point, about 2000 quadrupoles along each linear collider are needed to obtain a vertical beam size of 1 nm at the interaction point. Active mechanical stabilization is required to limit the vibrations of the magnetic axis to the nanometre level in a frequency range from 1 to 100 Hz. The approach of a stiff actuator support was chosen to isolate from ground motion and technical vibrations acting directly on the quadrupoles. The actuators can also reposition the quadrupoles between beam pulses with nanometre resolution. A first conceptual design of the active stabilization and nano positioning based on the stiff support and seismometers was validated in models and experimentally demonstrated on test benches. Lessons learnt from the test benches and information from integrated luminosity simulations using measured stabilization transfer functions lead to improvements of the actuating support, the sensors used and the system controller. The controller electronics were customized to improve performance and to reduce cost, size and power consumption. The outcome of this R&D is implemented in the design of the first prototype of a stabilized CLIC quadrupole magnet.

INTRODUCTION

In CLIC, electrons and positrons are accelerated in two linear accelerators to collide at the interaction point with an energy up to 3 TeV [1]. In order to reach the required luminosity – a measure of collision brightness – the beam size in the interaction point needs to be 1 nm in the vertical plane and 45 nm in the horizontal plane. This small beam size is achieved by focusing the beam with about 4000 Main Beam Quadrupoles (MBQs) in the main beam accelerator. There are 4 types of quadrupoles ranging from Type 1 with a length of 420 mm (100 kg) up to Type 4 with a length of 1915 mm (400 kg). Each quadrupole offset in relation to the beam, causes the beam size to grow at the interaction point, reducing luminosity. Offsets are introduced through quadrupole misalignment, ground motion and external forces on the magnet due to water cooling in the magnet, tunnel ventilation, etc. The static misalignment of the quadrupoles is reduced by an alignment system based on eccentric cams [2]. Two main mitigation techniques are used to reduce the effect of quadrupole vibrations on the luminosity. The first mitigation technique uses beam-based orbit feedback which measures the position of the beam with Beam Position Monitors (BPMs) and redirects it with dipole magnets. An alternate solution would be to reposition some of the quadrupoles between pulses (every 20 ms) to the nanometre level. The beam-based orbit feedback reduces the effect of the quadrupole's vibrations on the luminosity under 1 Hz and at multiples of the repetition rate of the beam (50 Hz).

The second technique reduces the vibrations of the quadrupole locally for each quadrupole by means of an active stabilization system which, along with the nano-positioning, is the subject of this paper. From beam dynamics simulations, a first estimate was made whereby the integrated root mean square (r.m.s.) for the power spectral density of the vibrations should not exceed 1.5 nm at 1 Hz vertically and 5 nm at 1 Hz laterally [3]. A vibration isolation system based on stiff piezo actuators has already been built, using commercial seismometers, reaching the required level [3][4]. This paper presents the existing and improved control systems for stabilization and positioning, their effect on the mechanical system and the influence of the accelerator environment on the global control scheme. Firstly, the configuration of the original stabilization controller based on the seismometer, a new system using an inertial reference mass as well as the controller for the positioning system is explained. Secondly, the constraints on the mechanical design due to the chosen control system are defined. Thirdly, the effect of the accelerator environment on the global control scheme and its lay out are described. Finally, the achievements of the different controller layouts in terms of the stabilization system's transmissibilities are revealed.

VIBRATION ISOLATION AND POSITION CONTROL SYSTEM

A stiff system based on piezo actuators was selected for the stabilization and positioning of the quadrupoles due to the expected external forces on the quadrupole magnet coming from the accelerator environment (tunnel ventilation, water cooling, interconnections of vacuum tubes,etc.). The quadrupole on the piezo actuators is represented as a 1 degree of freedom system (see Fig. 1). The equation of motion in the Laplace domain is given by

$$X(s) = \frac{k}{ms^2+k}W(s) + \frac{1}{ms^2+k}F(s) + \frac{k}{ms^2+k}\Delta(s) \quad (1)$$

with m being the mass of the quadrupole, k the spring stiffness of the active support and F the forces induced on the

* The research leading to these results has received funding from the European Commission under the FP7 Research Infrastructures project EuCARD, grant agreement no.227579

† Stef.marten.johan.janssens@cern.ch

magnet by water cooling and other direct forces. The variable Δ represents the elongation of the actuators depending on the controller used. The different control configurations are described in the following paragraphs.

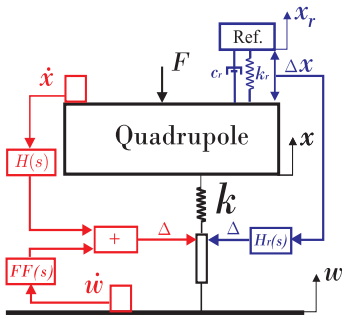


Figure 1: Schematic representation of an active isolation system with a piezo actuator.

Seismometer

The existing vibration isolation system uses commercial seismometers which measure the velocity of a magnetic reference mass with a coil. They have a bandwidth between 33 mHz and 100 Hz limited by several high order filters and are used in a feedforward/feedback configuration to reduce the vibration of the quadrupole. The elongation of the actuators is given by $\Delta = H(s)sX(s) + FF(s)sW(s)$. The feedback controller H(s) includes an integrator, a double lag to limit the bandwidth, a high-pass filter and the sensitivity curve of the seismometer. The feedforward controller FF(s) consists of a high and low-pass filter, an integrator and the seismometer sensitivity. The control system's main limitation is the feedback loop's instability caused by one set of poles coming from a high-order low-pass filter in the seismometer.

Inertial Reference Mass

A new sensor is under development which uses the relative displacement Δx between a reference mass ($X_r(s)$), with a suspension frequency of 1 Hz and a damping ratio $\xi_r = 30$ % achieved by actively damping the system, and the quadrupole position ($X(s)$) (see Fig. 1). This sensor will improve the stability of the controller as the additional poles of the seismometer are removed. Using a capacitive gauge or optical sensor as a measurement device in the inertial reference mass, allows it to be more suitable for an accelerator environment with stray magnetic fields than a seismometer using a coil. The elongation of the actuator for this configuration is given by $\Delta = -H_r(s)(X(s) - X_r(s)) = -H_r(s)X(s)(1 - G_r(s))$ with $G_r(s) = \frac{X_r(s)}{X(s)} = \frac{c_r s + k_r}{m_r s^2 + c_r s + k_r}$. The sensitivity curve looks like an under-damped high-pass filter.

Nano-Position Control

The nano-positioning is performed by a controller using the error ($e(s)$) between the requested quadrupole position ($R(s)$) and the actual relative quadrupole position ($Y(s) = X(s) - W(s)$). The actuator elongation is then given by $\Delta(s) = C(s)e(s)$. The controller $C(s)$ consists of a Proportional-Integral Controller (PI Controller). The limitation of the controller comes from the pole of the first mode of the system. In order to increase stability, a low-pass filter at 40 Hz is added.

MECHANICAL DESIGN CONSTRAINTS

The main limitations for the three different control systems come from instabilities in the feedback loop. The maximum performance of a feedback system is a function of the maximum feedback gain g at which the feedback loop becomes unstable. The margin of the actual gain to this maximum gain is called the Gain Margin (GM). The GM of the system is influenced by the poles related to the first mechanical mode in the system. In order to define the minimal mechanical characteristics of the system, the GM of the controllers with a fixed gain ($g = 10$ for the stabilization and $g = 105$ for the positioning) is set out to a range of first modes ($f_1 = \frac{1}{2\pi}\sqrt{\frac{k}{m}}$) in Fig. 2.

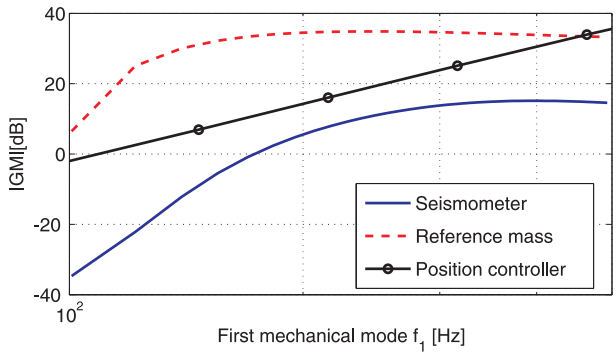


Figure 2: Stability margins for a vibration isolation system with a seismometer, a reference mass and the positioning controller, all with a fixed gain, in function of the first mode of the system.

It shows that the control system with the inertial reference mass has a higher gain margin than the commercial seismometer. Further, for both the inertial mass and the nano-positioning, the first mode should only be higher than 120 Hz in order to have sufficient performance in comparison to 250 Hz for the commercial seismometer. This input from the control system has been taken into account for the mechanical design in the form of an xy-guidance system to increase stiffness in the lateral direction [4] and puts requirements on the design of the alignment system on which the stabilization system will be mounted.

CONSTRAINTS ON GLOBAL THE CONTROL SCHEME

Working in a long linear particle accelerator brings its own set of challenges to the global control scheme for the local stabilization and position controller. The control centre is located kilometres away from the magnets that need

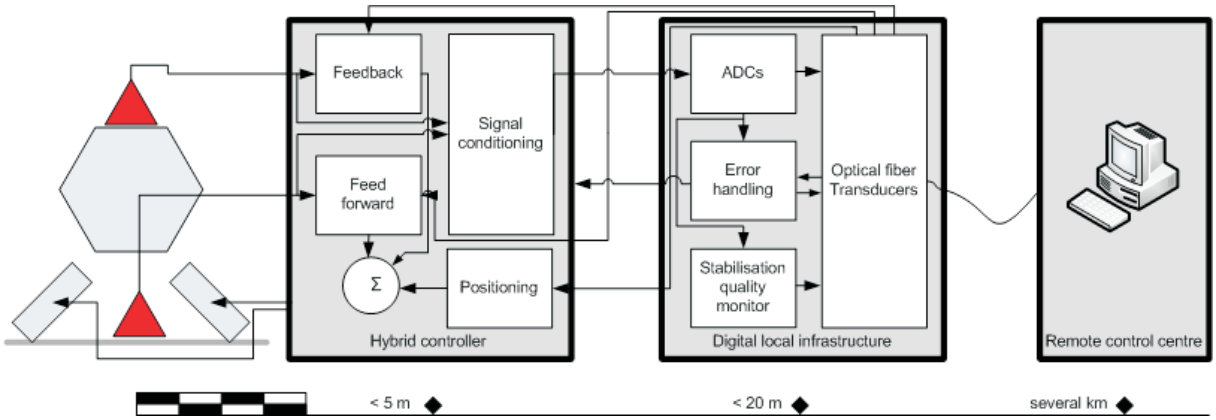


Figure 3: System level description of stabilization controller within CLIC control.

Table 1: Classification of Signals According to Timing Requirements and Direction

	Critical latency	Best-effort delay
Input	$\dot{x}$ or Δx	Self check
	$\dot{w}$	Emergency stop
	y	New position R
		Configuration parameters
Output	Δ	$\dot{x}$
		Error signal
		RMS vibration level of Magnet
		Performance figure

to be stabilized and positioned. It was demonstrated that the phase shift introduced by such a long communication path would be disastrous for the performance of the stabilization controller even when optical fibres are used [6]. To this end, the signals going into and coming out of the global control system are divided between delay critical and best effort delay signals (see Table 1). The signals for the stabilization and positioning controllers are time critical. This limits the distance allowed from the magnet to these controllers. However, the tunnel is an electromagnetically noisy environment and the sensor cables are a weak link. At the same time, putting the electronics closer to the magnet increases the radiation exposure. As a compromise, the stabilization and position controllers are put in a hybrid controller at a distance from the magnet in the order of metres as is shown in Figure 3. The final location will depend on civil engineering constraints. The hybrid mixed-signal electronic board is described in Ref. [5]. It is based on a low noise analogue circuit filtering, shaping, processing and generating the relevant control signals ($\dot{x}$ or Δx,$\dot{w}$ and Δ), and a digital part for the configuration of controller parameters (gain g, filter limits,...). A hybrid solution was chosen over a more classical digital construction using ADCs and DSPs or FPGAs for a number of reasons. ADCs with a resolution of at least 18 bits are required [6], which are not commercially available for radiation environments like CLIC. Latency was also shown critical for the stability and performance of the controller. To achieve the required delay, a very high sampling rate, without buffer, and hardware clock frequency is required. In addition, digital electronics are more sensitive to radiation, especially single events, than analogue electronics. The analogue cumulative errors are worse than those of a digital system but they can be partially compensated by gain corrections. All analogue components are chosen to have low noise and especially a low 1/f corner frequency, as the signal bandwidth of interest starts at 100 mHz (tantalum capacitors, metal thin layer resistors, etc.). Additionally a custom integrated control board has a lower cost, less power consumption, and reduced volume compared to a classic digital platform.

The best-effort delay signals include position, configuration parameters and status. They are processed at a local digital infrastructure which communicates with the remote control centre kilometres away. Only best-effort delay can be expected as these signals need to travel the distance to the remote control centre so a delay is unavoidable which is not a problem as long as the delay is kept constant.

SIMULATIONS AND TEST RESULTS

The effect of the ground vibrations on luminosity depends on the transmissibility of the vibrations from the ground to the quadrupole, the shape of the ground motion at that location and the beam-based orbit feedback, which works only below 1 Hz and at multiples of 50 Hz. Simulations for the transmissibilities of the local vibration controller were performed for a 100 kg mass (Type 1 magnet) on top of two piezo actuators resulting in a natural frequency of around 300 Hz in the vertical direction. Several different local controller configurations were simulated with both commercial seismometers and inertial reference mass (see Fig. 4). These seismometers were used in a feedback and feedback combined with feedforward configuration. The latter was tested on a prototype test bench, using a prototype of the hybrid controller, performing very close to the simulations. The transmissibility of this configuration was included in luminosity simulations [8] which showed that the peak at 80 Hz is undesirable as this is a location where the beam-based orbit feedback does not perform well. Using the reference mass reduces the performance locally at 7 Hz but gives a broader bandwidth as well as re-

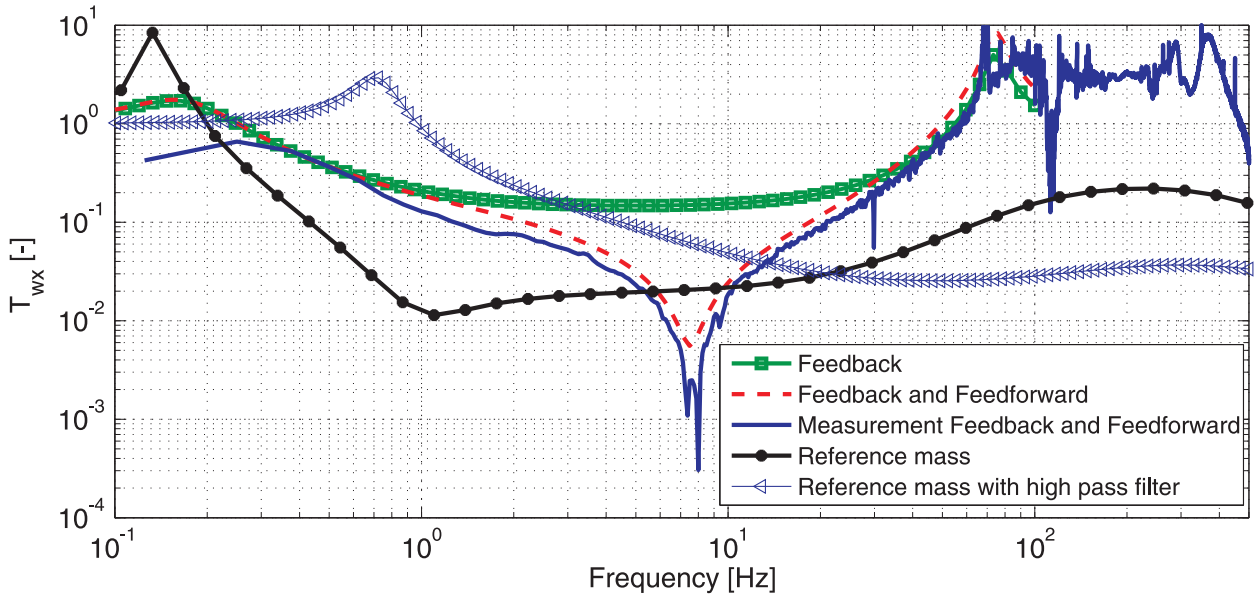


Figure 4: Transmissibilities between ground and quadrupole for the theoretical and measured feedforward combined with feedback system using a seismometer and the feedback using an inertial reference mass.

moving the unwanted peak at around 80 Hz. However, this configuration has a peak at around 0.2 Hz coinciding with the high ground motion due to incoming sea waves known as the micro seismic peak. Including a high-pass filter in order to move the low frequency peak away from the micro seismic peak resolves the problem. The transmissibility for this system also tested in a simulation of the whole accelerator including other mitigation techniques (beam-based feedback, etc.) with a ground motion model corresponding to the vibrations expected in the CLIC tunnel. It was found that the reference mass with the high pass filter gave the best performance reducing the luminosity loss from 68% to only 3 % although it is not the highest performing stabilization system at lower frequencies [7]. This shows that the interaction with the other mitigation techniques and local vibration levels are an important factor in defining the controller for stabilizing CLIC's main beam quadrupoles.

CONCLUSIONS

This paper has shown the effect of the different control systems on the mechanical system and the effect of the global control scheme due to the accelerator environment underlining that the first mechanical mode needs to be above 120 Hz for the position controller and if an inertial mass is used. The commercial geophones need a first mechanical mode higher than 250 Hz to keep it from destabilizing the stabilization controller in feedback configuration. Furthermore, the stabilization and position controller must be placed metres away from the magnet because of a performance drop due to latency and sensitivity to radiation and electromagnetic noise. The choice of a hybrid analogue/digital controller also contributes to reduce the latency. A digital connection was made which communicates with the remote control centre in order to keep some flexibility in the system for local changes in vibration levels and enable monitoring. A list of the interface signals was defined based on the critical delay and best-effort delay requirements. From beam simulations of the whole CLIC accelerator it was revealed that the shape of the stabilization controller is also defined by the interaction with the other mitigation techniques.

REFERENCES

[1] J.P. Delahaye,"Towards CLIC feasibility", IPAC' 10, Kyoto, May 2010, FRXCMH01, p. 4769.

[2] H. Mainaud Durand *et al.*,"Validation of a Micrometric remotely controlled pre-alignment system for the clic linear collider using a Test Setup with 5 degrees of freedom", IPAC' 11, San Sebastian, September 2011, MOP0029.

[3] C. Collette *et al.*, "Nano-motion control of heavy quadrupoles for future particle colliders: An experimental validation", Nuclear Instruments and Methods in Physics Research A, 643,(2011), pp. 95-101.

[4] K. Artoos *et al.*, "Status of a Study of Stabilization and Fine Positioning of CLIC Quadrupoles to the Nanometre Level", IPAC'11, San Sebastian, September 2011.

[5] P. Fernandez Carmona *et al.*, "Study of the hybrid controller electronics for the nano-stabilization of mechanical vibrations of CLIC quadrupoles", TWEPP' 11, Vienna, September 2011.

[6] K. Artoos *et al.*, "Study of the electronics architecture for the mechanical stabilization of the quadrupoles of the CLIC linear accelerator", TWEPP' 10, Aachen, September 2010, Jinst 5 C11014.

[7] S. Janssens *et al.*,"System control for the CLIC main beam quadrupole stabilization and positioning", IPAC' 11, San Sebastian, September 2011, TUPC014.

[8] J. Snuverink *et al.*, "Status of Ground Motion Mitigation Techniques for CLIC", IPAC' 11, San Sebastian, September 2011, TUPC023.

UPGRADE OF THE SERVER ARCHITECTURE FOR THE ACCELERATOR CONTROL SYSTEM AT THE HEIDELBERG ION THERAPY CENTER

J.M. Mosthaf*, K. Höppner, S. Hanke, S. Stumpf, A. Peters, T. Haberer
HIT, Heidelberg, Germany

Abstract

The Heidelberg Ion Therapy Center (HIT) is a heavy ion accelerator facility located at the Heidelberg university hospital and intended for cancer treatment with heavy ions and protons. It provides three treatment rooms for therapy of which two using horizontal beam nozzles are in clinical use and the unique gantry with a 360ï¿½ rotating beam port is currently under commissioning. The proprietary accelerator control system runs on several classical server machines, including a main control server, a database server running Oracle, a device settings modeling server (DSM) and several gateway servers for auxiliary system control. As the load on some of the main systems, especially the database and DSM servers, has become very high in terms of CPU and I/O load, a change to a more up to date blade server enclosure with four redundant blades and a 10Gbit internal network architecture has been decided. Due to budgetary reasons, this enclosure will at first only replace the main control, database and DSM servers and consolidate some of the services now running on auxiliary servers. The internal configurable network will improve the communication between servers and database. As all blades in the enclosure are configured identically, one dedicated spare blade is used to provide redundancy in case of hardware failure. Additionally we plan to use virtualization software to further improve redundancy and consolidate the services running on gateways and to make dynamic load balancing available to account for different performance needs e.g. in commissioning or therapy use of the accelerator.

THE HIT ACCELERATOR FACILITY

The Heidelberg Ion Therapy Centre (HIT) is a dedicated hadron accelerator facility for radio-therapeutical treatment of tumor patients [1, 2]. The two horizontally fixed treatment rooms as well as the gantry and experimental area (see Fig. 1) can be served in multiplexed operation with proton and carbon beams with qualified beam parameters (called MEFI, see Table 1), other ions like helium and oxygen have been tested.

The achieved energy range of 88-430 MeV/u for carbon ions and 48-221 MeV/u for protons is sufficient to reach a penetration depth of 20-300 mm in water. Patient treatment in the two horizontal treatment rooms is running at approximately 40 patients a day and the experimental area is used during night shifts. The gantry commissioning is ongoing and expected to finish in early 2012 [3, 4].

* joerg.mosthaf@med.uni-heidelberg.de

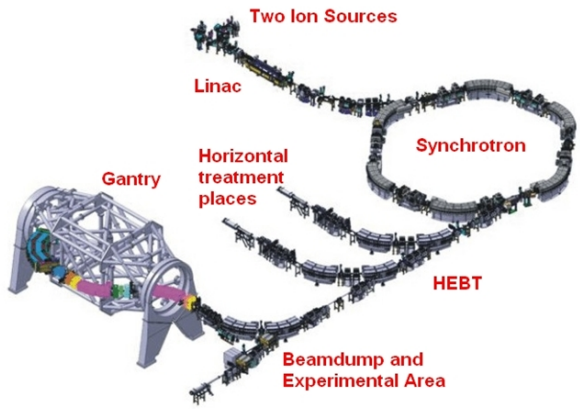


Figure 1: The HIT accelerator facility.

Table 1: MEFI Values

Parameter	Steps	Protons	Carbon
Energy	255	48–221 MeV/u	88–130 MeV/u
Focus	4(6)	8–20 mm	4–12 mm
Intensity	10(15)	$4 \cdot 10^8 - 1 \cdot 10^{10}$	$1 \cdot 10^7 - 4 \cdot 10^8$
Gantry Angle	36	365	365

SERVER ARCHITECTURE OF THE HIT ACCELERATOR CONTROL SYSTEM

The accelerator control system of the HIT facility (Fig. 2) was planned around several classical servers comprising the main ACS servers and the gateway and secondary servers [5, 6]. At the time of conception, this was the most practical way to limit negative influence of secondary systems and DSM calculations on the accelerator cycle.

Original ACS Servers

The original main ACS servers were housed inside two standard 19" racks together with secondary servers and gateways (see Fig. 3). A third 19" rack contains reserve servers and a network attached storage server (NAS) as well as a backup tape library. We used Fujitsu-Siemens TX100S2 and TX200S3 servers with dual core processors and 2 GB of memory running Windows Server 2003 and Oracle 9. This was deemed sufficient for the ACS and ran from 2005 through the commissioning of the facility until 2010 when gantry commissioning began in earnest. With the addition of the gantry angle as parameter for DSM calculations, the database tables for device data were filling rapidly. The addition of a second set of RAM data for de-

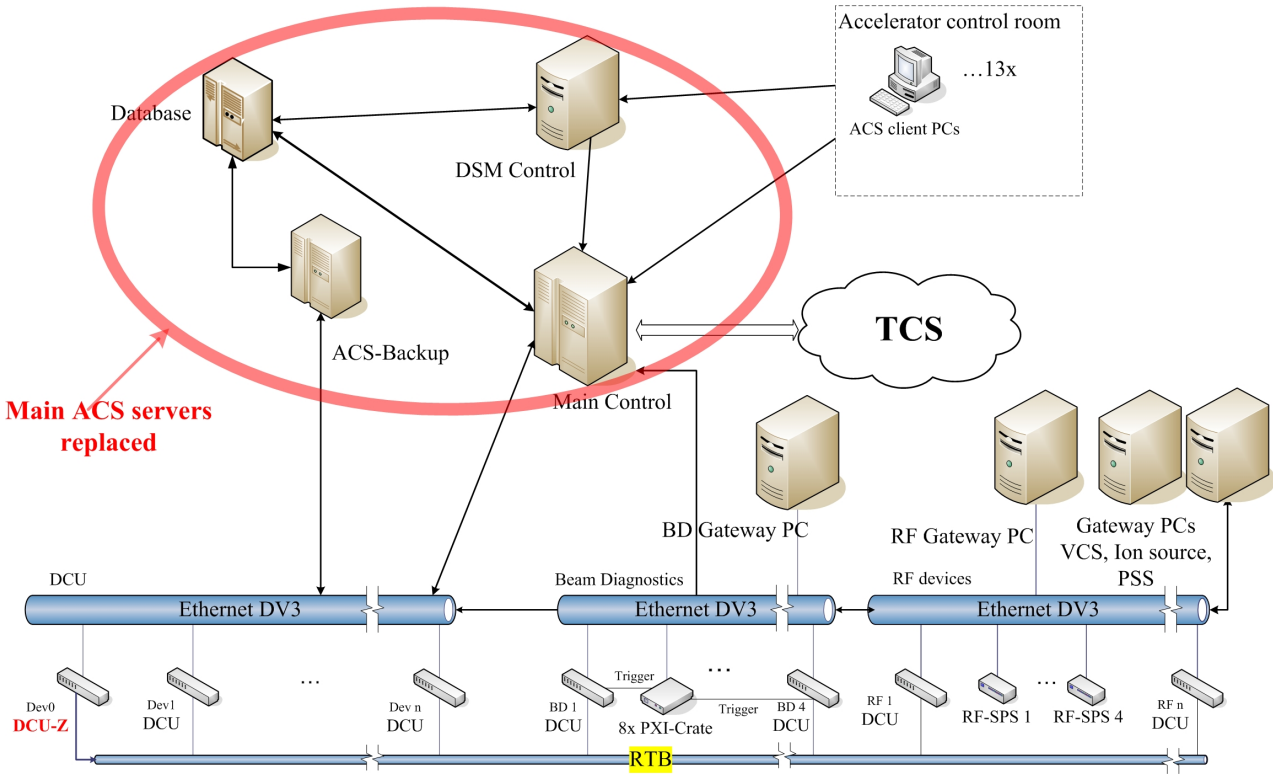


Figure 2: Server architecture and devices of the HIT accelerator control system with replaced servers shown.

vice control units to help in commissioning also strained the database. Underlying flaws in database design and configuration as well as limitations in CPU power, RAM and I/O speed reduced performance of operations and calculations of device settings. Regular statistic optimization and manual pruning were necessary to keep the database from choking.

Figure 3: One of the ACS 19" racks.

One complete interpolation of all gantry devices took between 4 and 6 hours per ion type. Downloading to the device control units (DCU) took more than 38 minutes and flashing took another 30 minutes. Cycle overhead (time between end of one beam cycle and start of a new one) was on average 1000 ms with a high spread. This proved to be an impediment mainly for commissioning of the gantry, but cycle performance was also an important consideration because of its influence on patient throughput. A performance analysis of the database system in cooperation with the industrial supplier of the control system (Eckelmann AG) and the GSI Helmholtzzentrum für Schwerionenforschung revealed issues in the database design as well as installation and configuration problems of the oracle server [7].

New Blade Servers

A part of the solution to these performance problems was a plan to replace the aging server structure with a new, state of the art blade center. Blade centers have several advantages over classical servers.

- High density - more processing power in less space
- Flexibility, modularity, and ease of upgrading - take out and add in server blades while the system is up and running
- Power consumption and power management - consolidation of power supplies and reduction of overall power consumption

- Network and other cabling - simplified cabling requirements and reduced wiring

- Load balancing and failover - blades have simpler and slimmer infrastructure and are designed for this task from the manufacturer

The disadvantages of blade centers are mainly high initial costs for the enclosure and vendor lock-in. To spread the costs over several budgets, it was decided to first replace only the main ACS servers more or less directly with blade servers and also procure the necessary infrastructure for further expansion. We started with one 16-space enclosure with redundant power supplies, management modules and dual Flex-10 10Gb/s Ethernet connections. Four identical blade servers with two 8-core CPUs, 24 GB of RAM and two internal HDDs and one storage blade with four HDDs in a RAID configuration are integrated into the enclosure (Fig. 4. The new database server, running Oracle 11 on Windows Server 2008, is connected to the storage blade and has a completely new configuration according to the tests and recommendations by GSI. All blades are interconnected and share the same power supply, management modules and network connection. After installation and configuration of the blade enclosure and the blade servers, the newest version of oracle 11g was installed on the designated database blade (blade 1 and storage blade) and the HIT ACS on the other servers. As planned, the second blade replaced the maincontrol server, the third blade the DSM and the fourth blade the ACS backup server. The third and fourth blade also double as backup blades in case of hardware failure.

Figure 4: The blade enclosure.

PERFORMANCE IMPROVEMENTS

Following the take-over of the new blade system for the old ACS servers, several measuring shifts were used to determine the performance gains. We compared pre-blade data of cycle times to several testing plans run in patient mode. These testing plans showed an average cycle overhead of 740 ms as opposed to ∼1000 ms. Scatter plots of cycle times before and after the server upgrade are shown in Figures 5 and 6. Red data points show cycles collected with the old servers, while green data points show cycles running with the new blade center. These points show a mix of different accelerator modes and so the decrease in average cycle overhead is not as pronounced.

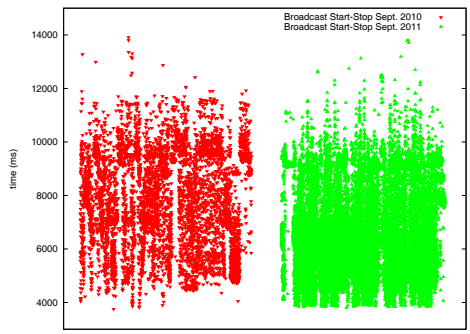


Figure 5: Scatter plot of cycle time.

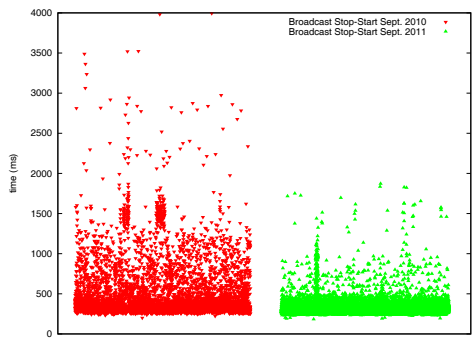


Figure 6: Scatter of cycle overhead.

The spread of the data points is smaller, especially in the cycle overhead graph. The better database performance results in less waiting time during cycle overhead and so reduces average overhead by approximately 30% in test plans. The average overall length of the cycle has been reduced also, while beam time is identical. The improvements in DSM calculation is very significant. As Table 2 shows, the interpolations and download have improved significantly.

Also very noticeable are the improvements in GUI performance. All GUI functions using database queries have been sped up by an appreciable amount. Most responses that took several seconds before, are now instantaneous. Only some functions, like reading and filtering longer log histories or loading data for therapy protocols, still take several seconds.

Table 2: Approximate Improvements with New Blade System

	Original	Blade	Gains
Interpolation (incl. gantry devices)	∼4–6 hrs.	∼1.5 hrs.	>200%
Download (per ion type)	∼40 mins.	∼10 mins.	∼400%
Flash (per ion type)	∼30 mins.	∼8 mins.	>250%
Avg. cycle overhead (with test plans)	∼1000 ms	∼740 ms	∼35%

CONCLUSIONS AND OUTLOOK

The performance gains with the new blade servers are substantial. The Table 2 shows great improvements in DSM computations and gains in operating are also significant. GUI performance is significantly improved and waiting times for database queries are vastly reduced. Even cycle overhead has shortened by a factor of ∼30% and is expected to improve more with new database enhancements. Redundancy is enhanced by the hot spare blade servers which are pre-configured to run all ACS services and by the dual Ethernet connection.

The next step in our upgrade will be to procure more blade servers and incorporate the auxiliary and gateway servers into the blade center. Also planned is a Storage Area Network (SAN) blade that will be connected to all blade servers via internal 10 Gb/s Ethernet and allows saving of server images and fast failover capabilities. More and dedicated hot spare blades will also bring more redundancy. We also plan on using virtualization software to run some of our servers with resource sharing and load balancing to better utilize our server resources to the fullest.

ACKNOWLEDGMENTS

The authors would like to thank Eckelmann AG for their support with the control system migration and testing and measuring of the cycle times as well as the GSI Helmholtzzentrum für Schwerionenforschung for their database performance analysis and their support with the database installation.

REFERENCES

[1] H. Eickhoff et al., "HICAT - The German hospital-based light ion cancer therapy project", EPAC'04, Lucerne, Switzerland, http://www.jacow.org.

[2] Th. Haberer et al., "The Heidelberg Ion Therapy Center", Radiotherapy and Oncology, Vol. 73 (supplm. 2), p. 186-190, 2004.

[3] A. Peters et al., "Operational Status and Further Enhancements of the HIT Accelerator Facility", Proc. IPAC10, Kyoto, Japan, http://www.jacow.org.

[4] M. Galonska et al., "Commissioning of the Ion Beam Gantry at HIT", Proc. IPAC11, San Sebastian, Spain, http://www.jacow.org.

[5] J.M. Mosthaf et al.,"Overview of the communication structure of the HIT accelerator control system", Proc. PCAPac'08, Ljubljana, Slovenia, http://www.jacow.org.

[6] T. Fleck et al., "Status of the Control System for the Therapy Facility HIT", Proc. PCAPac'08, Ljubljana, Slovenia, http://www.jacow.org.

[7] K. Höppner et al., "Improvement of the Oracle Setup and Database Design at the Heidelberg Ion Therapy Center", Proc. ICALEPS11, Grenoble, France, http://www.jacow.org.

A DIGITAL BASE-BAND RF CONTROL SYSTEM*

M. Konrad†, U. Bonnes, C. Burandt, R. Eichhorn,
J. Enders, N. Pietralla, TU Darmstadt, Darmstadt, Germany

Abstract

The analog RF control system of the S-DALINAC has been replaced by a new digital system. The new hardware consists of an RF module and an FPGA board that have been developed in-house. A self-developed CPU implemented in the FPGA executing the control algorithm allows to change the algorithm without time-consuming synthesis. Another micro-controller connects the FPGA board to a standard PC server via CAN bus. This connection is used to adjust control parameters as well as to send commands from the RF control system to the cavity tuner power supplies. The PC runs Linux and an EPICS IOC. The latter is connected to the CAN bus with a device support that uses the SocketCAN network stack included in recent Linux kernels making the IOC independent of the CAN controller hardware. A diagnostic server streams signals from the FPGAs to clients on the network. Clients used for diagnosis include a software oscilloscope as well as a software spectrum analyzer. The parameters of the controllers can be changed with Control System Studio.

We will present the architecture of the RF control system as well as the functionality of its components from a control system developers point of view.

INTRODUCTION

The S-DALINAC is an 130 MeV recirculating electron linac that is operated in CW mode. It uses superconducting niobium cavities at 2 K with a loaded Q of $3 \cdot 10^7$ for acceleration. Their 20 cell design and the high operating frequency of 3 GHz make them very susceptible for microphonics. In addition, superconducting 2 and 5 cell capture cavities, one of them supporting acceleration at velocities $\beta < 1$, are used inside the injector.

Furthermore, room-temperature chopper and buncher cavities are operated. A new polarized electron injector has been assembled in the accelerator hall [1]. Its bunching system consists of a chopper cavity and a 3 GHz as well as a 6 GHz harmonic buncher. Therefore the RF control system has to deal with different loaded quality factors (Q_L) ranging from some 5000 to $3 \cdot 10^7$ as well as with different operating frequencies.

The target specification for the stability of the accelerating field is an error in phase of $0.7°$ rms and a relative error in amplitude of $8 \cdot 10^{-5}$ rms. Since the old analog RF control system never achieved these values and became more and more unreliable it has been replaced by the new digital RF control system last year.

* Work supported by DFG through CRC 634.
† konrad@ikp.tu-darmstadt.de

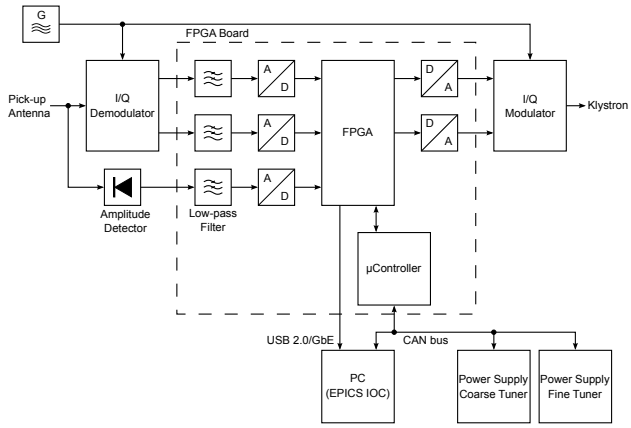


Figure 1: Overview over the hardware components of one channel of the RF control system.

CONTROLLER-BOARD HARDWARE

The RF control system converts the RF signals down to the base band. This allows to split the hardware into two parts: A frequency dependent RF board containing the I/Q (de)modulator and a frequency independent FPGA board processing the signals (see Fig. 1). All hardware components have been developed in-house. A separate power detector located on the RF board improves the accuracy of the magnitude measurement.

The analog signals are low-pass filtered on the FPGA board before they are digitized to avoid aliasing effects. The filters used for this purpose are third-order π filters with an edge frequency of $\approx$ 100 kHz. This filtering step is very important because the $19/20\,\pi$ mode of our 20 cell cavities is only separated by 700 kHz from the π mode used for acceleration. The chosen filter design ensures that the adjacent mode is suppressed by 55 dB while on the other hand the phase shift introduced by the filter is negligible for microphonic frequencies (up to 10 kHz).

High-linearity 18 bit ADCs are used to meet the specification in respect to accuracy. They run at their maximum sampling rate of 1 MS/s to keep the latency low and to allow the system to detect the signal of a detuned cavity operated in self-excited loop mode.

The FPGA used on the controller boards is a Xilinx Spartan 6 (XC6SLX45). Roughly 25% of its resources are used up to now leaving enough room for future extensions of the algorithm.

The requirements concerning the resolution of the DACs are not as high as for the ADCs. The DACs that are used on the FPGA board have a resolution of 16 bits and run with a frequency of 1 MS/s.

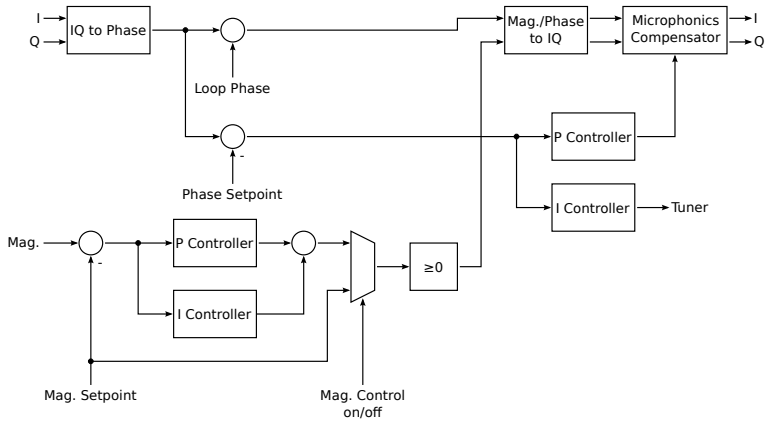


Figure 2: Simplified flow-chart of the self-excited loop control algorithm used with the superconducting cavities.

FPGA AND CONTROL ALGORITHM

The different Q_Ls of our cavities make different control algorithms necessary. The room-temperature copper resonators are operated with a generator-driven resonator (GDR) algorithm whereas for the superconducting cavities a self-excited loop (SEL) algorithm is better suited. Figure 2 shows a simplified flow-chart of the SEL control algorithm. In addition to simple operations like multiplication and adding operations it uses CORDIC blocks to transform from Cartesian coordinates (I and Q) to polar coordinates (magnitude and phase) and back [2]. For a more detailed description of the control algorithms see [3] and [4].

The control algorithm is implemented as a program for a self-developed soft CPU that is placed into the FPGA. The CPU is highly optimized for the implementation of RF control algorithms. Its arithmetic logic unit (ALU) uses 18 bit operands in order to fully exploit the accuracy of the ADCs. To keep things as simple as possible the instruction set is reduced to the operations needed for RF control algorithms. The CPU does not use complex memory access but only variable and parameter registers. Parameter registers are used for control parameters that are adjusted by the operator. They are read-only for the soft CPU but read/write for the micro-controller which connects the controller board to the PC. Variable registers used for intermediary results on the other hand are read-only for the micro-controller but read/write for the soft CPU. This concept together with a two-staged pipeline allows a high clock rate of 80 MHz. Complex operations like the conversion from Cartesian coordinates into polar coordinates and back are based on Xilinx CORDIC IP cores [5] that can be used from the CPU and that are pipelined as well (about 30 stages).

The CPU provides several 36 bit accumulating registers that can be used for the implementation of integral controllers. Only the 18 most significant bits are used for further processing but all 36 bits are considered for the accumulated sum. This allows integral controllers with time constants of roughly 1 s.

To avoid discontinuities all operations are available in a normal and a clipping variant. The latter clips the signal to the maximum/minimum 18 bit value if the result of an operation exceeds the range. The normal variant of the instructions is typically used for phase operations whereas the clipping variant is used for calculations with magnitude or I/Q values.

Using a soft CPU for the control algorithm has advantages in diagnostics because all variables (and thereby all intermediary results) are stored in fixed registers (block RAMs) where they can be read out easily. In particular there is no need for 18 bit data paths and huge multiplexers throughout the whole FPGA that would consume a great deal of the resources. In addition to this the soft CPU approach speeds up the development of the control algorithms. No time consuming synthesis of the Verilog code is necessary after the control algorithm has been changed. On the other hand the serial processing of the data reduces the performance. It turned out that for most practical purposes this penalty is small since the latency between ADC and DAC is determined mainly by the CORDIC blocks. The latency caused by the complete control algorithm is approximately 1 μs.

SLOW CONTROL

The controller boards are connected to a standard PC server via CAN bus. This interface is used to monitor and adjust all slowly varying (e. g. by user interaction) parameters of the control algorithm and the hardware. The micro-controller that connects the FPGA boards to the CAN bus runs Nut/OS, an open source real-time operating system providing cooperative multi-threading [6]. In addition to communicating with the PC it sends commands directly to the fine and coarse tuner power supplies via CAN bus. A three-step controller implemented in the micro-controller activates the motor tuner if the fine tuner approaches its limits.

The PC runs Linux and an EPICS IOC that is hooked up to the CAN bus via a device support that uses the SocketCAN network stack included in recent Linux kernels (since 2.6.25) [7]. SocketCAN provides access to the CAN bus

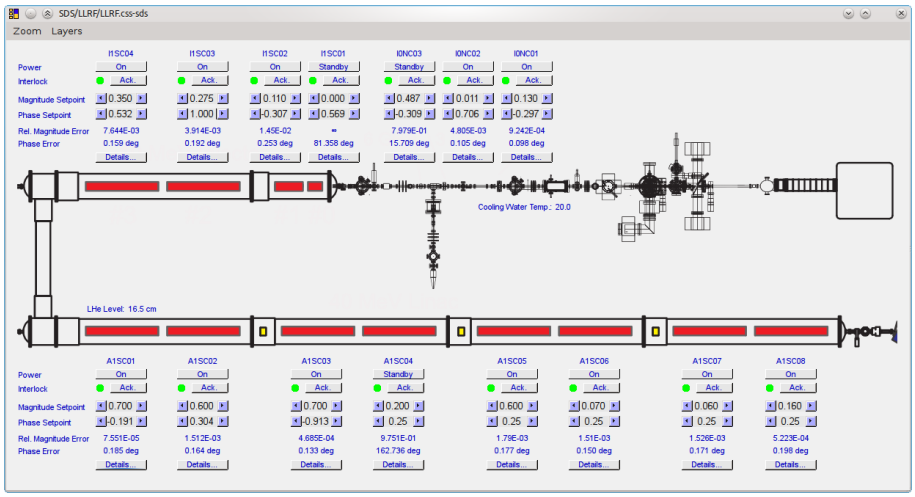


Figure 3: Screenshot of the main display of the RF control system's operator interface. It only provides the most frequently used controls and links to further displays for details.

via network devices (BSD sockets) which can be accessed by multiple applications at the same time (e. g. the IOC and a CAN sniffer). Furthermore it acts as an abstraction layer that makes the device support independent of a specific CAN card or hardware vendor (all CAN cards having a SocketCAN driver are supported).

All together the IOC provides several thousand records for all 16 RF channels. A great deal of them is related to diagnostics.

The operator interface has been implemented with Synoptic Display Studio [8]. Figure 3 shows a screenshot of a display providing an overview over all control loops and their most important controls. Further controls are available on control algorithm specific displays that contain all parameters for the selected controller.

DIAGNOSTICS

The FPGA boards provide fast diagnostic data via USB 2.0. Eight signals with a resolution of 16 bits are continuously transferred into the PC with the full sampling rate of 1 MS/s. The two least significant bits of the full 18 bit result are available as separate channels.

During operation eight controller boards are plugged into a crate together with a concentrator card. The concentrator card uses an FPGA to collect the data from the controller boards and streams the selected signals to the PC over its own USB 2.0 interface (cp. Fig. 4). Two crates can be coupled over a parallel LVDS interface. In this configuration one concentrator module acts as master and the other as slave. Signals from different controller boards and crates can be transferred at the same time.

In addition to the eight channels with full sampling rate the master concentrator card provides a stream of all 64 signals of all 16 controller boards on a second USB 2.0 interface. This data is transferred with a reduced sampling rate of 2 kHz and is intended for monitoring. Both USB data streams end at the PC where the IOC reads the data via a specially developed device support.

The data streams are forwarded by the diagnostic server process to the connected clients via TCP connections. Typical client applications include a software oscilloscope [9] that has been extended with a data source plug-in that reads data over the network from the IOC as well as a software spectrum analyzer [10]. Data recording is possible with standard Unix tools (e. g. netcat).

In addition to forwarding the stream of data the slow data stream is fed into EPICS records by the device support. It provides mean values of all signals every 0.1 s which can be used for monitoring.

Those values can be used to detect if a controller is out of lock but they do not provide precise information about the performance of the cavity and its controller. Even the oscilloscope view of the fast streaming data is not very helpful because it is difficult to see small changes in a noisy signal. A much more meaningful measure for the errors of the field in the cavity are the RMS errors of the magnitude and phase error signals. They could be calculated from the fast streaming data at the PC but this would be possible only for four cavities even if all eight diagnostic channels are sacrificed for this purpose. To cover all cavities while keeping the diagnostic channels free for the operator the RMS calculation has been moved partly into the FPGAs of the controller boards. They have the whole data stream available and can relieve the PC from processing work by carrying out the most time-consuming part of the calculation. The FPGAs calculate the square of the signals for each ADC sample and sum 2^{17} of them up which corresponds to a time of roughly 0.1 s. The result is transferred to the PC via CAN bus. The EPICS IOC calculates the square root of the sums and scales the values to finally get an error in degrees or a relative error in case of the magnitude respectively.

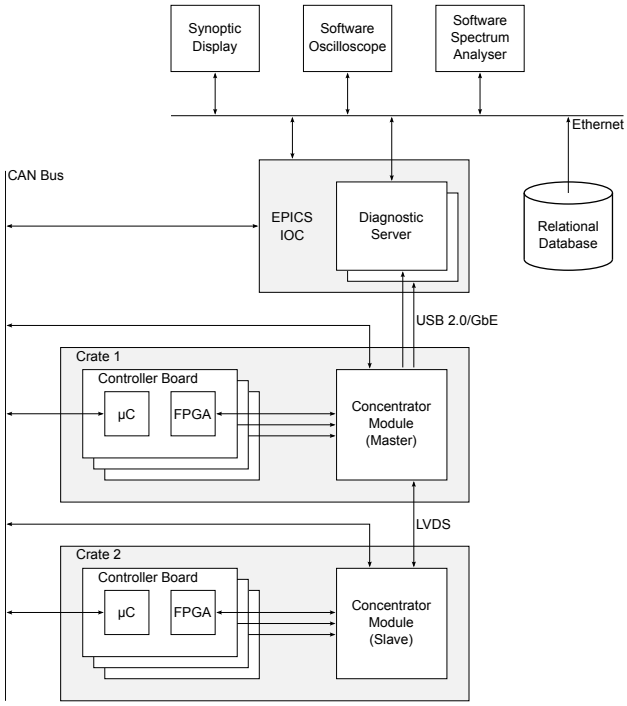


Figure 4: Data flow between controller boards, concentrator modules, diagnostic server and clients.

SOFTWARE AND FIRMWARE DEPLOYMENT

The RF control system has been put into operation as soon as the major components allowed stable operation. A lot of bug fixes and new features have been integrated since then. It turned out that the concept of continuous integration works best for us. It requires to recompile and deploy the software to the PC server and firmware to the microcontrollers and FPGAs frequently and – most important – in a repeatable fashion. This has been achieved by using the Debian Fully Automatic Installation (FAI) [11] to setup the PC server over the network. A virtual machine is used for testing to make sure all packages are installed cleanly before the production machine is reinstalled. To speed up the installation some software components are automatically build by a build server and distributed as Debian packages. This way changes can be deployed to the test or the production system within minutes. A complete installation of the server takes about 10 minutes. This means that downtimes of the accelerator of half an hour can be used to upgrade the complete RF control software. Updates of the IOC are possible even during operation because the controller boards run completely independent of the PC.

The IOC configuration is generated automatically from a relational database and a set of templates during installation. This way we can recover quickly from a broken PC without the need to hold completely installed and configured spare hardware available. Instead one spare machine is sufficient for all PCs running IOCs.

The micro-controller firmware images of the FPGA boards and the tuner power supplies can be updated via CAN bus. A boot loader makes it possible to recover from broken firmware images without physical access to the JTAG chain. The configuration of the FPGAs can be written using the same technique. A multi-boot functionality provides a fall-back to a second "golden" image for the FPGA in case of a broken primary bitstream [12].

These measures together with revision control of all components make a significant contribution to reduce the downtime of the RF control system.

SUMMARY

The digital RF control system is based on in-house developed hardware giving us full control over every detail. The soft CPU that has been implemented in the FPGA allows fast and flexible modification of the control algorithm on one hand and powerful diagnostics on the other hand. The availability of extensive diagnostics has proven to be a considerable precondition for the smooth commissioning of a new RF control system.

A highly automatized deployment of all parts of the software and configuration allows a fast integration of further improvements into the production system.

REFERENCES

[1] C. Eckardt et al., "The S-DALINAC Polarized Injector SPIN – Performance and Results", Proceedings of PAC 2011, New York, USA.

[2] J. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic Computers, pp. 330-334, September 1959.

[3] M. Konrad et al., "A Digital Low Level RF Control System for the S-DALINAC", Proceedings of LINAC 2010, Tsukuba, Japan, p. 268.

[4] M. Konrad et al., "Design and Commissioning of a Multi-Frequency Digital Low Level RF Control System", Proceedings of IPAC 2011, San Sebastián, Spain, p. 433.

[5] Xilinx Inc.: "LogiCORE IP CORDIC v4.0 – Product Specification", http://www.xilinx.com/support/documentation/ip_documentation/cordic_ds249.pdf.

[6] Ethernut. The Ethernut project. http://www.ethernut.de.

[7] SocketCAN. The SocketCAN project. http://developer.berlios.de/projects/socketcan/.

[8] K. Kasemir: "Control System Studio Applications", Proceedings of ICALEPCS 2007, Knoxville, USA, p. 692.

[9] Osqoop. The open oscilloscope. http://gitorious.org/osqoop.

[10] SigBlips: Baudline. A time-frequency browser. http://www.baudline.com.

[11] FAI. The Debian Fully Automatic Installation project. http://fai-project.org.

[12] Xilinx Inc.: "Spartan-6 FPGA Configuration – User Guide", http://www.xilinx.com/support/documentation/user_guides/ug380.pdf, pp. 123-128.

LHC SUPERTABLE

M. Pereira, T.E. Lahey*, M. Lamont, G.J. Müller, D. D. Teixeira, CERN, Geneva, Switzerland
E.S. McCrory, Fermilab, Batavia, USA

Abstract

LHC operations generate enormous amounts of data. This data is being stored in many different databases. Hence, it is difficult for operators, physicists, engineers and management to have a clear view on the overall accelerator performance. Until recently the logging database, through its desktop interface TIMBER, was the only way of retrieving information on a fill-by-fill basis. The LHC Supertable has been developed to provide a summary of key LHC performance parameters in a clear, consistent and comprehensive format. The columns in this table represent main parameters that describe the collider's operation such as luminosity, beam intensity, emittance, etc. The data is organized in a tabular fill-by-fill manner with different levels of detail. Particular emphasis was placed on data sharing by making data available in various open formats. Typically the contents are calculated for periods of time that map to the accelerator's states or beam modes such as Injection, Stable Beams, etc. Data retrieval and calculation is triggered automatically after the end of each fill. The LHC Supertable project currently publishes 80 columns of data on around 100 fills.

MOTIVATION

Even before the LHC came into operation it was already evident that there would be a need for means of easily obtaining general statistics on the performance of the accelerator. The LHC has been in stable activity for several months now and a large amount of data has already been produced. The increase of accelerator's performance and the thirst of physicists for data highlighted the need for a tool that processes and displays information on performance and other statistics.

Although the data is commonly available and accessible through dedicated tools it is troublesome to create an overall view of the operation of the LHC. One of the main data sources is the LHC Logging Service [1]. This service consists of two databases where data from all LHC devices and parameters is stored. Access to these databases is done through a Java API or through a desktop application called *TIMBER*. While the Java API allows other developers to retrieve data programatically, TIMBER can provide plots and data export into different formats. Although these tools fulfill their intended purpose they are much better at providing data of specific aspects of the LHC than producing the full picture. Another limitation of the available data mining tools available is that their use is typically restricted within CERN.

* On leave from SLAC, Menlo Park, California, USA

As for the Tevatron [2], an application was developed to provide summary data of the performance of the LHC.

THE SUPERTABLE

The supertable is a spreadsheet which provides a summary of LHC key parameters that can be used to assess the accelerators performance [3]. The columns of this table represent beam or accelerator parameters, such as, luminosities, beam intensities, crossing angles, filling scheme, etc. Each row contains the values of these parameters for each fill. The intent of the LHC Supertable is not to provide detailed data on each of these parameters. For that purpose, specialized and dedicated tools have been developed over the years and are used by experts on each of the domains. Instead this application aggregates and formats data in a concise and clear way so that physicists, experts and management alike can be given a general overview of the collider. In summary the LHC Supertable aims at:

- Providing a summary of the key parameters that detail the performance of the LHC.
- Providing a concrete mechanism for determining these key parameters on which everyone can agree (or, at least, through which the parties that disagree can have a basis for their argument).
- Displaying data in easily understandable formats such as web pages and excel sheets.
- Being a small, simple and concise resource.

A similar system has been implemented in Fermilab for the Tevatron accelerator. Fermilab's Supertable has been in operation since 2004 and currently stores data on thousands of fills. In order to profit from Fermilab's experience with their implementation, the LAFS workgroup was called in to give their input and contribution to the project. As a result, CERN's Supertable architecture is greatly inspired on Fermilab's experience [4]. A proposal was made [5] to implement a web application that would gather data from various data sources, process and store in a dedicated database and finally publish the digested data. The solution would profit from Fermilab's experience while the implementation would take in account the specificities of the LHC.

IMPLEMENTATION DETAILS

The LHC Supertable is a web application developed in Java and using the Spring framework. The application is divided in two components: persistence and web.

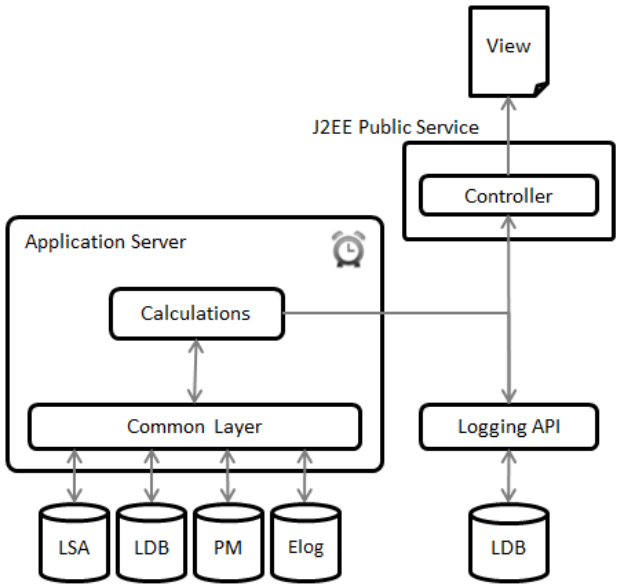


Figure 1: Simplified architecture overview of the LHC Supertable application.

Persistence Module

The persistence module is responsible for gathering data from the various data sources, processing the data and persist it in the database. This process is triggered by a cron job scheduled to run every 30 minutes. When the process is started there is first a verification if there are new fills to be calculated. If so, the *processor* will iterate through the columns of the supertable, calculating them one by one for that particular fill. The algorithms themselves are encapsulated in Java classes called *cells*. Each *cell* in the Java code is responsible for the calculation of one or more supertable cells. In example, the beta star scheme in the supertable is calculated within the *BetaStarSchemeCell* Java class, while luminosities are calculated in the *LuminosityCell*. With this separation of concerns each *cell* becomes an independent unit responsible for processing a set of columns in the table. Figure 1 shows a simplified view of the architecture.

Each attribute in the Supertable has an associate algorithm description and unit. These two elements are of great importance as beam or accelerator parameters may be calculated in different ways. However, the algorithm used to calculate a certain property may change over time. The Supertable offers the possibility to change algorithms inbetween fills or recalculate past fills. The calculation of parameters, i.e. peak ATLAS luminosity, typically involves the retrieval of data from one of the data sources. The Supertable is currently gathering data from the Logging Database, LSA [6], Postmortem [7] and the E-logbook. The current implementation is flexible enough to support the introduction of new data sources. Since one of the primary motivations of the Supertable is for a simple, clean and organized presentation the columns have been classified into one or more specific categories, which we call

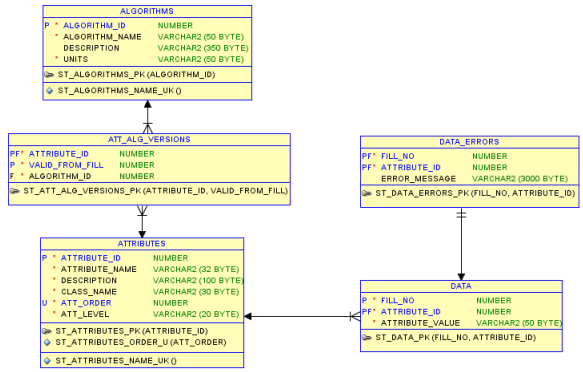


Figure 2: Implemented database schema.

views. The user can select one of these simple views to focus his/her attention on his/her area of interest. For example, the Luminosity view will only show columns concerning luminosity. The organization of columns into *views* provides different levels of detail according to the user's needs.

Data Storage

After all columns have being calculated the data is persisted in the database. For that purpose a schema has been created in the Logging Database. The Logging Database ensures the persistance of the data for the living time of the LHC. Hence, for the sake of consistency and centralization of services, the data of the Supertable was stored in the same environment as the rest of the logging data.

As shown in Figure 2 the database schema is quite simple but provides a lot of flexibility in the supertable. The table ATTRIBUTES is where the columns of the Supertable are defined. It currently holds 87 different attributes and, as mentioned before, each one has a class name field which corresponds to the name of the Java class responsible for calculating the value of the attribute. Additionally, the ordering and level of the columns are also defined in this table allowing to easily change the appearance of the Supertable in the web interface. Add, remove and edit operations on attributes can be performed via a dedicated administration package. This data driven approach makes the structure more flexible making it unnecessary to redeploy the application module responsible for the reads and writes in the database. In the ALGORITHMS table are defined both the algorithm and units per attribute of the Supertable. It was decided not to separate algorithms and units in different tables since they are strongly coupled and the separation wouldn't bring a significant added value in terms on flexibility. Since it is possible to change the algorithm being used in the calculation of a Supertable column without having to recalculate past fills an extra table was needed. The ATT_ALG_VERSIONS table keeps track of which algorithms were used to calculate each attribute throughout the calculated fills. Whenever a new algorithm is added

for the calculation of a column, this table will contain the fill number from which the algorithm went into effect. In case errors occurred at the time of the calculation of a fill these are stored in the data errors table. These errors should be displayed to the user to explain the reason behind eventual missing values for some columns. The choice of the logging database as final location for this scheme brought along an inherent limitation. Due to how the backup process occurs in the logging database, data becomes read only after a undefined period of time. In practice this means the recalculations of fills can only be performed provided that the backup process hasnt handled those particular fills. This schema doesn't currently offer any kind of versioning of data but we believe that this could be one of the solutions to workaround the mentioned constraint. Data from the Supertable can be programatically accessed through the Logging API using the *SupertableController*.

Web Module

The second module of the LHC Supertable is the web interface. The technology chosen was Google Web Toolkit. GWT allowed the creation and maintenance of a powerful Javascript front-end developed 100% in Java. The interface was made clean and as simple as possible but providing enough functionality to be useful. It displays data in tabular form and provides the algorithms used to calculate each parameter as well as the units. Columns are organized in 5 views and by default only a small subset will be displayed. In addition, exports in Excel and JSON are available allowing users to use data as they see fit.

CURRENT STATUS & OUTLOOK

The project as specified has been fully implemented as shown in Figure 3. The LHC Supertable currently publishes 87 columns of data for over 300 fills. Several views are available providing more or less detail according to the users needs. Data can be exported in both excel and JSON [9] formats along with a static html version. Despite the implementation of all planned functionalities the Supertable suffered from a great problem early on: quality of data. For some of the trickiest parameters to calculate algorithms were adopted from Fermilab's Supertable such as emittances, luminosities from emittances, among others. However, some of the implemented algorithms didn't translate well to the LHCs reality. Even though some of those algorithms have been changed, some errors still persist in the calculations. At the same time the LHC Supertable project was being developed another project was under way. The LHC Performance and Statistics although similar in goal started from a different point.

The Statistics application focused at first on creating a repository on ROOT files containing data on a set of key performance parameters of the LHC during the course of a fill. These files are meant to be used by physicists or experts who want to explore in more detail what happened

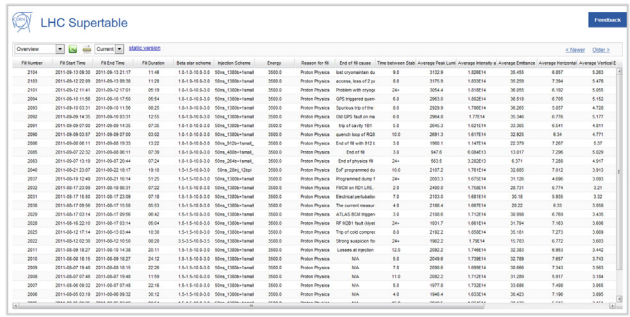


Figure 3: LHC Supertable web interface.

during a particular fill. Additionally, the Statistics application also made available a set of ROOT generated plots and a small table describing a limited subset of parameters also characterized by the Supertable. After some analysis of the functionality and architecture of both applications it was clear that these weren't concurrent systems but rather complementary. Hence a decision was taken to merge the two systems and provide a better service to the users of both applications. At the time of this writing the developer teams of both projects are reviewing the calculation algorithms used for the LHC Supertable which will be later applied in the new application. A system that provides statistics on the operation of the LHC is very much needed and of the utmost importance, however, it is rendered useless if it provides erroneous data.

CONCLUSIONS

The LHC Supertable has been designed, implemented and commissioned to provide a clear and consistent view of LHC operations data to all interested parties in the CERN community. As with any complex project, this product has evolved over its short lifetime to adapt to the environment in which it must exist. While input from the Tevatron Supertable has been invaluable, the LHC calculations often have been subtly different from the comparable Tevatron calculations, leading to some confusion in the customer base. As the user base grows it has become clear that this application is filling a void and providing functionalities that were not available in the existing expert tools. The merge with the LHC Statistics and Performance project is ongoing and should soon go into production. The final product will bring users a new set of functionalities and data for our users.

REFERENCES

[1] C. Roderick et al., "The LHC Logging Service: Handling Terabytes of On-line Data", ICALEPCS'09, Kobe, Japan, January 2010.

[2] T. Bolshakov et al., 'Data Acquisition and Analysis for the Fermilab Collider RunII', June 2004, FERMILAB-Conf-04/093-AD.

[3] D. McGinis, "Collider Supertable-Table for a Modern Hadron Collider", September 2008.

[4] Elliott McCrory, "The LHC Super-Table Project", June 2010, http://beamdocs.fnal.gov/AD/DocDB/0036/003628/002/Experience%20with%20Super%20Table.docx

[5] Elliott McCrory, "A Proposal for the Super-Table Generation Software", June 2010, http://beamdocs.fnal.gov/AD/DocDB/0036/003628/002/A%20Proposal%20for%20a%20Framework%20for%20Building%20the%20SuperTable.docx

[6] G. Kruk et al., "LHC Software Architecture [LSA] - Evolution toward LHC Beam Commissioning", ICALEPCS'07, Knoxville, USA, October 2007, WOPA03, p.307.

[7] Odd Oyvind Andreassen et al., "The LHC Post Mortem Analysis Framework", ICALEPCS'09, Kobe, Japan, January 2010

[8] Elliott McCrory, "Data Storage Schema for the LHC Super-Table, June 2010, http://beamdocs.fnal.gov/AD/DocDB/0036/003628/002/Data%20Storage%20Schema%20for%20the%20LHC%20SuperTable.docx

[9] Introducing JSON, http://www.json.org/

CONSTRUCTION OF NEW DATA ARCHIVE SYSTEM IN RIKEN RI BEAM FACTORY

M. Komiyama#, N. Fukunishi, RIKEN Nishina Center, Wako, Saitama, Japan
A. Uchiyama, SHI Accelerator Service, Ltd., Shinagawa, Tokyo, Japan

Abstract

RIKEN Radioactive Isotope Beam Factory (RIBF) is a cyclotron-based next-generation radioactive beam facility. RIBF uses two types of control systems; an experimental physics and industrial control system (EPICS)-based system and a group of several non-EPICS-based systems. For each control system there is a corresponding data archiving system in operation, and in order to unify these two data archives, since October 2009 we have been developing a new archiving system. This new data archive system, named RIBF Control Data Archive System (RIBFCAS), is required, at intervals between 1–60 s, to collect and store more than 3000 data generated by EPICS controlled devices through EPICS input/output controllers (IOCs). In addition, RIBFCAS must be able to combine the existing non-EPICS-based systems. To fulfill these requirements, a Java-based platform has been created, and client applications are based on Adobe AIR runtime. The RIBFCAS hardware system, therefore, consists of an application server, a database server and client-PCs. Presently, we have succeeded in stably acquiring approximately 3000 data from 22 IOCs every 10 s. Moreover, incorporation of the non-EPICS-based data archive system into RIBFCAS is now in progress.

OVERVIEW OF RIBF CONTROL SYSTEM

With an aim to explore unknown properties of unstable nuclei, the RIKEN Radioactive Isotope Beam Factory (RIBF) began operation in 2006 as the first of several next-generation RI beam facilities planned worldwide. RIBF is a cyclotron-based in-flight facility, and RIBF accelerators can supply RI beams at energies hundreds of MeV/nucleon over the entire range of atomic masses [1].

As shown in Fig. 1, the RIBF control system has a degree of complexity. The primary components of RIBF accelerators, such as magnet power supplies, beam diagnostic devices and vacuum systems, are controlled by an experimental physics and industrial control system (EPICS)-based system [2]. Additionally, there are several standalone control systems not integrated into the EPICS-based system, for example, those for radio frequency (RF) systems, the cooling water system and various long-standing ion sources. These systems operate independently of the remainder of the RIBF control system.

The EPICS-based system utilizes a number of device controllers, such as a Versa Module European (VME) control board, numerous types of programmable logic controllers (PLCs), general-purpose interface bus (GP-IB) controllers, PIC network interface card kit (PICNIC) [3], computer automated measurement and control (CAMAC) in-house controllers and in-house controllers based on network interfaces,. To integrate these device controllers within the EPICS framework, EPICS input/output controllers (IOCs) running "iocCore" software are necessary. "iocCore" is a set of EPICS routines that define process variables and implement real-time control algorithms. In our system, almost all EPICS driver/device supports for controllers can also be executed on Linux-based IOCs.

CAMAC in-house controllers, Device Interface Modules (DIMs), are managed by using the network crate controller CC/NET, a commercial product of Toyo Corporation [4, 5]. Because CC/NET is a Linux-based single-board computer, we can execute EPICS base software on it, and therefore the CC/NET itself becomes an IOC. In addition to CC/NET, a PLC-CPU module newly developed by Yokogawa Electric Corporation, F3RP61-2L (hereinafter, RP) [6], also works as an IOC; a soft real-time Linux environment. In contrast, the network-based in-house controller, Network-DIM (N-DIM) [7], the GP-IB controllers, PICNIC and the PLCs do not contain general operating systems (OS) on which to execute EPICS base software. To manage these controllers in the EPICS framework, they must be connected to additional IOCs through Ethernet connections to convert the communication protocols between EPICS channel access (CA) and the N-DIM or GP-IB controllers or PICNIC or PLCs. Linux-based IOCs, small single-board computers (ALIX) [8] are used to convert the protocols. The exception to the above is for NIO, a commercial control board from Hitachi Zosen Corporation, which is instead controlled by a device/driver support originating from VxWorks.

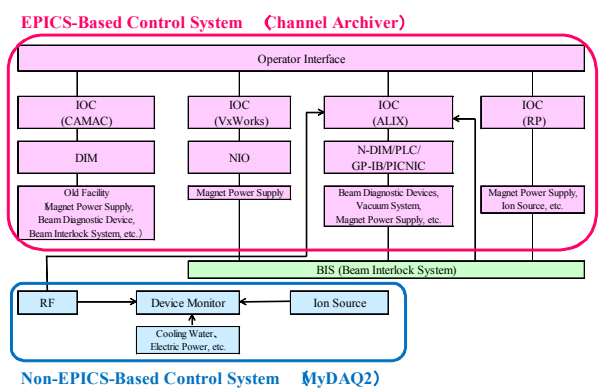


Figure 1: Schematic of RIBF Control System.

misaki@riken.jp

DIMs control approximately 370 magnet power supplies, and 100 beam diagnostic devices and vacuum systems. N-DIMs control around 250 beam diagnostic devices and vacuum systems. NIO boards control about 420 magnet power supplies. RPs control approximately 50 magnet power supplies and newly developed 28 GHz electron cyclotron resonance ion sources [9]. PLCs control the vacuum systems of cyclotrons and certain beam diagnostic devices. Although RIBF has many PLCs, EPICS controls only a proportion of these; Other PLCs have their own control systems and are independent of EPICS.

Two data archive systems have been used to record the various parameters of RIBF during operation. The pair "Channel Archiver" and "Archive Viewer", developed by the EPICS collaboration [10], is used as a data archiver for the EPICS control system. A new version of Channel Archiver, "RDB Channel Archiver", was released in 2010 to store data and configurations in a relational database (RDB); however, our system still uses the old release.

The second achieve system, "MyDAQ2" developed by the SPring-8 control group [11], is used to acquire and store non-EPICS-based systems parameters. Initially, each non-EPICS-based system individually managed their own data, and MyDAQ2 was introduced in order to manage all non-EPICS-based data in a cohesive manner. MyDAQ2 obtains data from each device through an Ethernet and stores them in a MySQL database such that users can retrieve stored data by using an associated Web application.

Although coexistence of two data archives covering different sections of the accelerator system does not cause significant problems in daily use, unifying the two database systems is favored. However, it is difficult to apply Channel Archiver to our non-EPICS-based systems. On the contrary, while it may be technically feasible to apply MyDAQ2 to our EPICS system, handling vast amounts of data is highly demanding. MyDAQ2 was developed for users who perform experiments at one of the SPring-8 beam lines, not the entire Spring-8 control system, and as a result the interface of MyDAQ2 is suitable for handling smaller amounts of data.

A drawback of the coexistence of two data archives is that data processing is necessary to investigate correlations in the stored data. For example, suppose we notice a gradual decrease in beam intensity, we are then required to investigate the correlation between operation parameters, such as temperature of cyclotron magnets and beam intensity. Since beam intensity data are stored in the Channel Archiver and magnet temperature data are stored in MyDAQ2, data processing is needed for comparison between the data. With a unified system, performing correlation analyses becomes easier and may be helpful for a more stable operation of RIBF. Furthermore, if the unified system can be used not only to display retrieved data but also to monitor real-time data, we can immediately fix an observed instability.

For the reasons indicated above, we decided to construct a new data archive system. This new system must meet the following requirements: acquire all parameters essential for the stable operation of RIBF accelerators at the required sampling rates, retrieve stored data and monitor real-time data.

DEVELOPMENT OF NEW DATA ARCHIVE SYSTEM

Outline

Since October 2009, we have been developing a new data archive system, RIBF Control Data Archive System (RIBFCAS), which covers both EPICS-based and non-EPICS-based systems. An outline of RIBFCAS is as follows:

1) Adoption of an open-source database was considered preferable.
2) For the EPICS-based system, a Java-encoded data acquisition program was newly developed to communicate with IOCs.
3) For the non-EPICS-based system, RIBFCAS works together with MyDAQ2 instead of developing new interfaces with the standalone control systems.
4) The client program has an ability to show both past and real-time data.
5) We plan to retain stored data for a minimum of five years. Data from up to two years previously can be searched immediately from client PCs at all times; however, older data are retrieved from a backup.

The RIBFCAS data acquisition system has to be capable of collecting more than 3000 data values at 1–60 s intervals, depending on the device type. System performance is highly reliant on the choice of database. Accounting for query performance, especially for large tables, we concluded that PostgreSQL 8.4.1 satisfied our fundamental requirements, because it supports a basic table partitioning function.

To achieve 2), a CA library must be used in RIBFCAS data acquisition program. The performance of this CA library is also important when storing the large data discussed above. A number of CA libraries were tested in order to find the most suitable for our intended purpose, and the results are discussed in the proceeding section.

For 3), MyDAQ2 data are merged with the data from the EPICS-based system by using a client application. We have created a CGI program to retrieve data from MyDAQ2 and export them in extensible markup language (XML) format to the RIBFCAS client application.

In realizing 4), a client application was developed on Adobe AIR. The advantage of using this platform is that client applications can be executed in multiple Adobe AIR runtime environments regardless of the file format.

Performance Test of CA Libraries

We evaluated data acquisition speeds for three CA libraries: Java Channel Access (JCA), Java Channel Access Light Library (JCAL) and an adapter library for JCA application programming interface (API) (JCAL–JCA). JCA is supported by the EPICS collaboration [12], and JCAL and JCA–JCAL were developed by the J-PARC control group [13]. Performance of the libraries was examined by writing programs with each such that all data could be processed for three chosen IOC types, having approximately 200 parameters apiece. One process of the test programs attempted to obtain data every 1 s and a test was concluded by obtaining five successive failures or successes. The JCA program was implemented for a single thread, whereas the JCAL program was implemented for multiple threads, since JCAL was developed as a thread-safe library in order to improve the stability of JCA when used in multi-thread technology. We measured the data acquisition time during each test and the results are summarized in Table 1.

The JCAL and JCA–JCAL programs successfully processed all data within 1 s, where the average time to acquire a datum was approximately 2 ms. There are no significant differences between the performances of JCAL and JCA–JCAL. When compared with these two programs, a longer data acquisition time was found for the JCA program due to the introduction of a delay time in order to establish a safe connection between the program and an IOC. If the JCA program attempts to process data from any channel of an IOC following a JCA-API call to establish channel connection, the program receives signals indicating an unconnected status and returns a failure message. To avoid this scenario, it was necessary to wait for 40 ms after each connection was established. For the case of the CC/NET IOC, this delay was increased 100 ms. As a consequence, we selected JCAL as the CA library for RIBFCAS.

Table 1: Performance of EPICS Java Library

IOC	Number of Parameters	JCA Execution Time (ms)	JCA Average (ms)		JCA–JCAL Execution Time (ms)	JCA–JCAL Average (ms)		JCAL Execution Time (ms)	JCAL Average (ms)	
CC/NET	187	49830	43	Fail	5102	2	Pass	5035	2	Pass
ALIX	198	46986	42	Pass	5104	2	Pass	5042	2	Pass
VME	192	45540	42	Pass	5101	1	Pass	5037	2	Pass

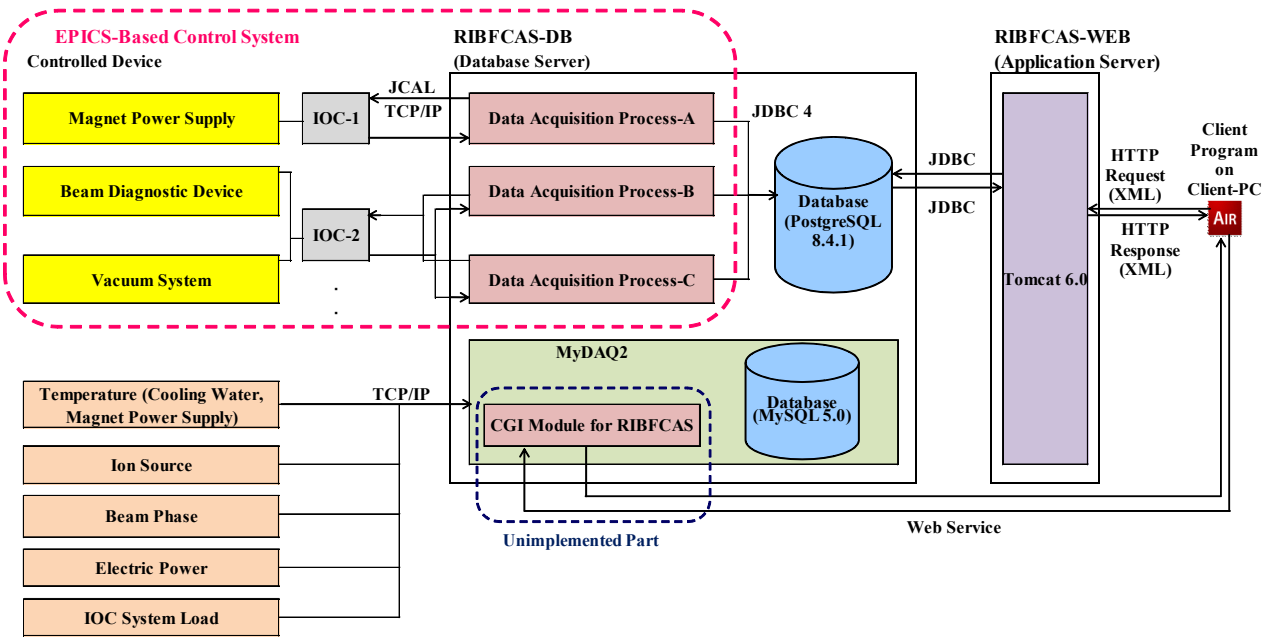


Figure 2: RIBFCAS Schematic Overview.

Structure of RIBFCAS

Figure 2 shows the structure of RIBFCAS. RIBFCAS consists of a database server, an application server, and client PCs. These are all connected to the EPICS local area network. The database server (RIBFCAS-DB) executes a JCAL-based data acquisition program to collect EPICS data. In addition, MyDAQ2 currently runs on RIBFCAS-DB and the MySQL database is also managed on this server. The application server (RIBFCAS-WEB) implements Tomcat version 6.0, which is a Web application server for the Java environment. Tomcat executes tasks in response to requests from client PCs, and returns retrieved data to client PCs in XML format. On client PCs, applications are executed on Adobe AIR runtime.

Table 2 shows the hardware specification of these servers and PCs. However, the performance of the existing database server is not sufficient to fulfill our requirements. Hence, we plan to upgrade the server.

The data acquisition program has the following features:

- The data collection program is divided into several processes and multi-thread technology is used to execute these processes simultaneously. This program structure gives the flexibility to divide

existing processes into segments or to add new threads for devices introduced to RIBF accelerator complex in future upgrades.

- A list containing a large amount of information collected by RIBFCAS is managed within the XML framework in the program, making it easier to change device data.
- The system has an organized multilayer structure; components such as the database, data collection processes and data collection logic in the database server, Web service and client services are isolated from each other.

The main RIBFCAS system was completed in FY2010 and performance tests commenced in FY2011.

Table 2: Hardware Specification of RIBFCAS

	RIBFCAS-DB	RIBFCAS-WEB
CPU	Intel Xeon (Quad Core) 2.33 GHz	Intel Core2Duo 2.20 GHz
Memory	16 GB	1 GB
Hard disk drive	120 GB (RAID10, SAS)	80 GB
OS	CentOS 5.2 (32 bit)	CentOS 5.3 (64 bit)

OPERATIONAL STATUS

A trial for the EPICS-based section of RIBFCAS began on 26 April 2011, according to the operating schedule of RIBF. The program acquires values for approximately 3000 various component parameters of RIBF from 22 IOCs by using nine separate processes. During the test period, RIBFCAS has been storing data of the vacuum systems of cyclotrons and beam transport lines, the magnet power supplies and the beam diagnostic devices, once every 10 s. Although the present data sampling rate of 10 s is sufficient to monitor the stability of the systems, it is also possible to optimize the time interval for acquisition depending on device type.

Thus, we have succeeded in continuous data acquisition without significant problems occurring. The stored data has been constantly growing and reached approximately 76 GB on 13 July. During this period, unexpected system interruptions have transpired on several occasions, with the data collection process terminated due to limitations in memory usage of the 32-bit OS. The cause of this phenomenon might be unresolved small memory leaks in the data acquisition program. To avoid these interruptions resulting from memory usage limitations, we have adopted a practice of restarting the relevant processes routinely and automatically. Note that this phenomenon was not observed for CC/NET.

The client application has also been shown to work well. The client program successfully monitors real-time data and can also retrieve data for any period stored in the database. In both cases, a user can select any amount of any data for viewing. The client program displays data over a 24 h interval for a parameter within 10 s, and data over 1 h within 0.3 s. Even more, only 0.05 s is required to recall 24 h data of a parameter once it has been referred to. Monitored or retrieved data are displayed as a time plot in the client application and obtained plots can also be saved as text or JPEG files.

FUTURE PLAN

Although investigation of the source of the memory leakage in the data acquisition program must be pursued, RIBFCAS has started to collect stably data from the EPICS-based system. As a next step, we plan to develop an additional client application that processes the data stored by MyDAQ2. This is an important stage in unifying the two existing databases. Another problem to be solved is management of the vast amounts of data. As already stated, the EPICS-based section of RIBFCAS collects 76 GB of data every 2.5 months, which corresponds to 400 GB of stored data every year. Selection of appropriate hardware to handle such data is now in progress. We plan to complete the development of RIBFCAS in FY2011.

ACKNOWLEDGMENTS

We would thank to Mr. T. Etoh and Mr. Y. Otaka in Swadok Co., Ltd. for their coding the RIBFCAS programs.

REFERENCES

[1] N. Fukunishi, et al., "Operating Experience with the RIKEN Radioactive Isotope Beam Factory", PAC09, Vancouver, Canada, May 2009, MO3GRI01, p.60 (2009)
[2] M. Komiyama, et al., "Upgrading the Control System of RIKEN RI Beam Factory for New Injector", ICALEPCS2009, Kobe, Japan, October 2009, TUP084, p.275 (2009)
[3] http://www.tristate.ne.jp/picnic-e.htm
[4] http://wwwonline.kek.jp/~yasu/Conference/rt2003/CAMAC/RT2003-CAMAC.pdf
[5] http://www-online.kek.jp/~yasu/Parallel-CAMAC/
[6] http://www.yokogawa.co.jp/rtos/Products/rtos-prdcpu9-ja.htm
[7] M. Fujimaki, et al., RIKEN Accel. Prog. Rep., 37, p. 279 (2004)
[8] http://pcengines.ch/alix.htm
[9] N. Nakagawa et al., Rev. Sci. Instrum. 79 02A327 (2008)
[10] http://www.aps.anl.gov/epics/extensions/index.php
[11] T. Hirono, et al., "Development of Data Logging and Display System, MyDAQ2", PCaPAC08, Ljubljana, Slovenia, October 2008, TUY01, p. 55 (2008)
[12] http://epics-jca.sourceforge.net/jca/
[13] H. Sako, et al., "Development of High-level Application Framework with a Script Language JCE for Accelerator Beam Commissioning", ICALEPCS2009, Kobe, Japan, October 2009, THP091, p. 842 (2009)

ALGORITHMS AND DATA STRUCTURES FOR THE EPICS CHANNEL ARCHIVER

J. Rowland, M.T. Heron, S.J. Singleton, K. Vijayan, M. Leech,
Diamond Light Source, Oxfordshire, UK

Abstract

Diamond Light Source records 3GB of process data per day and has a 15TB archive on line using the EPICS Channel Archiver. This paper describes recent modifications to the software to improve performance and usability. The file-size limit on the R-Tree index has been removed, allowing all archived data to be searchable from one index. A decimation system works directly on compressed archives from a backup server and produces multi-rate reduced data with minimum and maximum values to support time efficient summary reporting and range queries. The XMLRPC interface has been extended to provide binary data transfer to clients needing large amounts of raw data.

INTRODUCTION

This paper investigates the architecture of the EPICS Channel Archiver and describes the changes made to improve performance and usability at Diamond. Other database designs were also considered as part of the upgrade progress and for intermediate processing of decimated data, and this information is summarized as a reference for future archiver developments. Information about the data structures used in external storage is not always available in user documentation, but it is essential to consider these together with hardware capabilities and query patterns when good performance is required.

Problem Statement

The EPICS Channel Archiver (Archiver) records data from N channels, each producing samples at a different rate. We assume that timestamps are monotonic, and that samples arrive sorted in time but unsorted by channel. The goal of the Archiver is to manage this data and fulfil queries in a timely manner to support operations. The most common query returns samples from M << N channels between times T0 and T1, so query results exhibit the opposite locality of reference to sample insertion. If the most common query returned samples from all channels in a small time range then there would be no mismatch between insertion and retrieval and the problem would be trivially solved; for example a sequential logfile of samples with a sparse index of timestamps would suffice.

Locality of reference is important because it determines how to find the next sample in a query; sequential samples can be iterated without traversing an intermediate data structure. The problem is compounded when the storage medium penalises random access. This is the case for all common systems, as RAM, solid-state disks and hard disks transfer data in blocks. Hard disks have the extra problem of mechanical latency when positioning the head.

Channel Archiver at Diamond

The important figures for Diamond are presented in Table 1. The Archiver is split across multiple engines for functional isolation and to utilize multiple processor cores. Diamond also operates a redundant system with two complete Archiver servers and disks to ensure high availability, so every component in the table is duplicated.

Table 1: Archiver Parameters

Data Size	12.5 TB
Data Rate	3 GB/day
Index Size	23 GB
Archive Engines	39
EPICS Channels	148819
Server RAM	24 GB
Server CPU	2x Intel X5670
Server Cores	24 (HT)
RAID Storage	36 TB
RAID Cache	1GB (write-back)
RAID groups	5

DATA STRUCTURES

Chunked Arrays

An array contains a single type of element, and supports appends and random reads. An array is contiguous in storage so that element addresses can be calculated directly, but resizing involves allocating new storage and copying the whole array. Maintaining contiguity whilst allowing for growth is possible using the 'dynamic array' where the array is resized by a constant multiplicative factor rather than one element at a time. This gives amortized O(1) append complexity, and is commonly used for in-memory arrays such as the C++ std::Vector. Drawbacks are memory fragmentation, unpredictable delays due to copying, and the need for increasing amounts of space at the end of the array.

An alternative is the chunked array, where the array is not copied on resize and is no longer fully contiguous. Storage is allocated in chunks; now element addresses must be mapped to chunk locations using an index. This

is the storage mechanism used by the Archiver and other scientific data stores. The Archiver also chains chunks together in a linked list so that the index is only touched to find the initial chunk of a query. One common file format supporting chunked arrays is HDF5, used for beamline experimental data at Diamond [1].

The Archiver has a minimum chunk size of 64 samples. As new chunks are added to an array the chunk size is doubled up to a maximum of 8192 samples. This reduces wasted space for small arrays and at the end of chunks whilst ensuring that large arrays have reasonable runs of contiguous storage.

The Archiver also partitions array storage by date, into weekly data directories. Once written, partitions are immutable, making backups simple as old directories do not have to be scanned for updates.

R-Tree

The Archiver uses an R-Tree as the chunk index. An R-Tree is a sorted search tree optimized for efficient range intersection queries. Each tree node has M children; having large nodes reduces the depth of the tree and the number of seeks required to complete a search compared with a binary tree. At Diamond the default M value of 50 has been increased to 200 to reduce the depth of the Master Index.

The keys in the tree are timestamps and the values are chunk offsets. As the archiver does not allow overlapping chunks to be stored in an array, the index structure could be replaced with a B-Tree without loss of functionality. This is relevant as there are many robust B-Tree implementations available offering useful upgrades such as compression and file locking [2].

Hash Table

The Archiver contains many channels, so the first level of index is a chained hash table with linked lists mapping from channel names to chunk indices. The default table size of 1009 entries resulted in 150x overfilling for Diamond channels, and the linked list nodes are spread throughout the index file, making lookups expensive. An option was added to the ArchiveIndexTool to adjust the hash table size and the Master Index was rebuilt using a prime near 300000, improving channel retrieval performance.

Master Index

The Hash Table and R-Trees for an archiver partition are stored in an index file in the partition directory. The partitions are joined by merging the individual index files into the Master Index. The data type used to store file offsets was increased from 32 to 64 bits to allow the size of the Master Index to grow above 2GB. This increases usability by presenting the user with a single index for all data; previously the user was responsible for splitting queries across multiple indices. Also, the interpolation algorithms in the server only produce consistent bin boundaries when retrieval is from the same index.

HARDWARE

Diamond uses a magnetic disk system in a RAID5 configuration to store Archiver files. The Archiver services all queries from the on-disk arrays, so samples must be written soon after they are received to support timely control room diagnostics. Samples are stored in circular buffers in RAM, and written every 30s. This write operation appends samples to every chunk that has been updated in the last period, and chunks are scatted across the disks. This is illustrated in Fig. 1; numbers in the matrix are disk offsets; shading indicates write order. The write pattern is not contiguous.

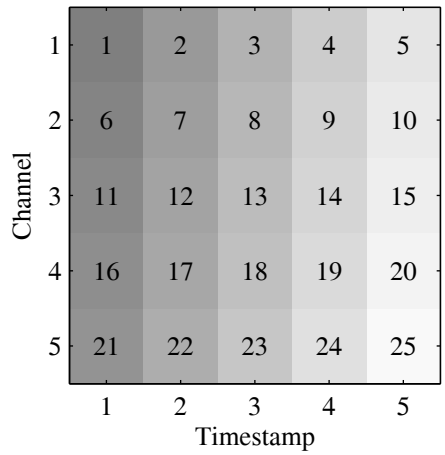


Figure 1: Archiver Write Order.

A typical magnetic disk can perform 100 I/O operations per second (IOPS) and has a sustained sequential transfer speed of 100MB/s. A disk seek costs as much as transferring 1MB of data. Each channel update costs one IOP. Therefore we can service approximately 3000 channel updates every 30s, assuming that none of the array chunks are sufficiently close together on the disk for their writes to be merged into a single IOP. This demonstrates that the key to Archiver performance is servicing write requests. RAID5 configurations typically have reduced random write performance due to the need to update parity bits, but spreading the archiver partitions across multiple RAID groups increases random write performance linearly with the number of RAID groups.

Write-Back Cache Controller

The RAID controller in the Diamond Archiver server has a battery-backed write-back cache with 1GB of RAM, enough to store 8 hours of data. Increasing the write period allows small writes to be merged, greatly increasing throughput. Figure 2 shows the write pattern after re-ordering by the cache controller.

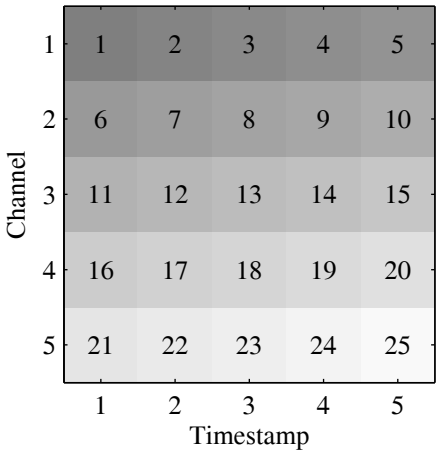


Figure 2: RAID Cache Write Order.

Table 2 shows the I/O utilization reported by the Linux tool IOSTAT when running the full Diamond archiving workload on three systems with different RAID controllers. 100% write utilization indicates that samples are discarded. Old is the previous production archiver, New is the current production archiver with upgraded cache controller, Desktop is a machine without a write-back cache controller.

Table 2: I/O Utilization

Hardware	Write%	Read%
New	3	75
Old	40	100
Desktop	100	100

RETRIEVAL

The Archiver query interface is exposed through the XMLRPC language-independent remote procedure call. This has proved pleasant to use from a variety of platforms including VB.NET, Matlab, Python and Java. However some users retrieve a large number of raw samples from the Archiver to produce multi-year reports and found the performance disappointing even after the server and disk upgrade. The XML encoding of arrays of EPICS datatypes was the limiting factor, these were packed as arrays of structures with the field names in ASCII for each sample, leading to a large space and time overhead. A new XMLRPC retrieval type was developed returning the EPICS sample array as a structure of arrays, one for seconds, nanoseconds, value, status and severity, each array packed into a byte buffer in native little-endian format and base64 encoded. Table 3 shows the performance improvement, the NFS method shows the performance of direct file access. The number of samples per request was still limited to 10000 as the RPC result is assembled in memory before sending to the client, this has since been increased to 1M samples to reduce the overhead of the RPC roundtrip.

Table 3: Retrieval Method Peformance

Method	Overhead	Time
NFS	1x	10m20s
XMLRPC	20x	112m7s
XMLRPC Base64	1.3x	26m

COMPRESSION

At Diamond backups are compressed with tar and gzip, and typically achieve a 4:1 compression ratio. Some of this is accounted for by partially empty chunks at the end of each array partition, but the low entropy of sorted EPICS time series data makes it an ideal candidate for compression. Figure 3 shows the sparsity pattern of the difference between adjacent samples for real double-precision data. High-order bytes of the timestamp corresponding to days and months, status and severity words, and padding bytes put in to enforce aligned memory access all rarely change.

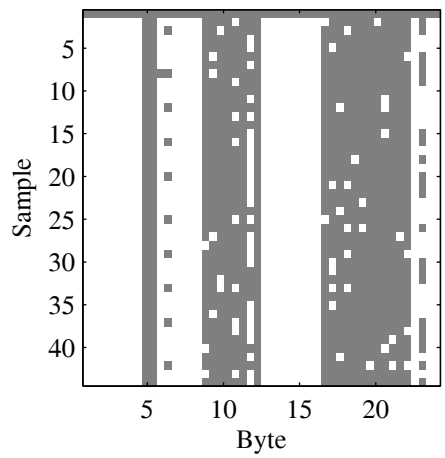


Figure 3: Differences of Adjacent Samples.

Compressed tar files are not seekable and cannot be read by the Archiver tools directly but libarchive [3] can stream decompressed data from a tar file without extracting to a temporary directory, this is used at Diamond to perform complete scans of the archive. Every sample can be read in a few days directly from compressed backups. This was used to feed the decimation tools discussed below.

Another option for direct access to compressed data is the squashfs [4] file system. This uses gzip in blocks to create a seekable compressed file system image that can be mounted by Linux. Using this to compress older data files will effectively quadruple the capacity of the Archiver storage without any software development effort. Squashfs is part of the Linux kernel from version 2.6.29.

DECIMATION

Many queries request decimated data from the archiver, and this decimation is performed on-line, traversing every raw sample on disk in the time range. As part of the upgrade program a method of pre-calculating these decimated queries was investigated. The required performance was achieved without making use of pre-decimated data, but the method is described because the data structures are more generally useful and offer an alternative to the chunked array store.

The data structures for decimated data were based on Hypertable [5], a write-optimized sorted key value storage engine developed to run on commodity hardware. Hypertable has previously been considered as a storage engine for the Archiver by INFN [6], and is itself inspired by the Google Bigtable [7] database.

Hypertable

Hypertable eliminates random writes through the use of a RAM cache. The key components of Hypertable are as follows:

- Sequential logfile for durability,
- RAM cache for data re-ordering (MEMTABLE),
- Periodic writes to immutable sorted disk tables (SSTABLE),
- Periodic external mergesort,
- Index of tables.

All operations are appended to a logfile which is periodically flushed to disk. In case of system failure, pending operations can be replayed from the logfile. Key and value pairs are maintained in a sorted data structure in RAM known as the MEMTABLE, and written to immutable sorted lists known as SSTABLEs (sorted string tables) once the RAM table is full. This provides similar functionality to the write-back RAID controller, but in software only. Keys are located in SSTABLEs by a B-Tree index.

In summary Hypertable provides a write-optimized structure to maintain partially sorted data on disk with the degree of sorting limited by available RAM. To improve sorting, SSTABLEs are periodically merge-sorted together. Typically the maximum size of the SSTABLE must be limited to prevent full-disk sorts. For EPICS time series data a suitable sort key is the (channel, timestamp) pair. This will ensure that channel samples are stored with good locality on disk for retrieval.

The decimation system uses a simple implementation of the Hypertable scheme. The sort key is (decimation rate, channel name, binned timestamp). The MEMTABLE uses the Berkeley B-Tree in-memory database. Raw Archiver samples are read from the compressed backups. If the MEMTABLE contains a binned sample containing the timestamp, the sample is added to the bin, otherwise a new bin is created. Once the MEMTABLE is full, it is saved to disk as a list of key value pairs. Once all samples have been read, the SSTABLEs are merge sorted into a single large table, and a sparse index is created containing the file offsets of each channel. The final table and index are compressed with squashfs.

Decimating all channels at 10 minute, 1 hour and 1 day intervals results in a compressed data size of 62 GB. An XMLRPC server provides the same interface as the original Archiver. Retrieval performance for long decimated queries is improved in proportion to the decimation interval.

SUMMARY

The required performance and usability improvements to the EPICS Archiver at Diamond were achieved without major changes to the software, but it was essential to consider hardware and software together to achieve this. Table 4 shows some benchmarks of the old and new systems. The DI benchmark returns decimated data for a week from a few hundred channels, the VA benchmark returns raw data for years from a few 10s of channels.

Table 4: Benchmarks

Hardware	Test	Time
New	DI	30s
Old	DI	3m30s
New	VA	8m30s
Old	VA	110m30s

For future Archiver developments the Hypertable combination of logfile and in-memory queryable cache offer a way of implementing an efficient and durable storage engine without requiring a hardware RAID controller.

REFERENCES

[1] HDF5, http://www.hdfgroup.org/HDF5

[2] Berkeley Database, http://www.oracle.com/us/products/database/berkeley-db

[3] http://code.google.com/p/libarchive

[4] http://squashfs.sourceforge.net

[5] http://hypertable.org

[6] Mauro Giacchini, Hypertable Archiver, http://www.lnl.infn.it/~epics/joomla/archiver.html

[7] http://labs.google.com/papers/bigtable.html

LHC DIPOLE MAGNET SPLICE RESISTANCE FROM SM18 DATA MINING

H. Reymond, O.O. Andreassen, C. Charrondiere, G. Lehmann Miotto, A. Rijllart, D.A. Scannicchio,
CERN, Geneva, Switzerland

Abstract

The splice incident which happened during commissioning of the LHC on the 19th of September 2008 caused damage to several magnets and adjacent equipment. This raised not only the question of how it happened, but also about the state of all other splices. The inter magnet splices were immediately studied and new measurements recorded, but the internal magnet splices were still a concern. At the Chamonix meeting in January 2009, the CERN management decided to create a working group to analyse quench data of the magnet acceptance tests in an attempt to find indications for bad splices in the main dipoles. This resulted in a data-mining project that took about one year to complete. This presentation describes how the data was stored, extracted and analysed reusing existing LabVIEW™ based tools. We also present the encountered difficulties and the importance of combining measured data with operator notes in the logbook.

INTRODUCTION

After the melting of a busbar interconnect splice in sector 34 in September 2008 during hardware commissioning, a special campaign was initiated to find bad splices by calorimetric and quench protection system (QPS) measurements [1]. These measurements were made in sectors 12, 56, 67, 78 and 81. The sectors 23, 34 and 45 were not measured, due to the fact that they were not cold at the time of the campaign.

No bad splices were found within the detection limits of 20 nΩ, but two bad magnet splices of about 100 and 50 nΩ were found inside the dipole magnets (MBBI334 and MBBI303). When these magnets were opened 'hardly soldered' splices were indeed found between poles and apertures.

The Chamonix Workshop (January, 2009) therefore gave a recommendation to look back in the magnet test facility (SM18) data and try to find indications for bad splices looking at the old provoked quench test data.

A working group was formed with members from the Technologies, Engineering and Physics departments to make this analysis [2]. It was decided to study with priority the magnets in sectors 23, 34 and 45, for which the least precise measurements existed, as they were not cold at the time of the campaign. It was also decided to study only the main dipoles. The main quadrupoles were connected in quite a different way in SM18 than in the LHC, thus making the joint measurements very uncertain.

During the tests, after repair of the damaged magnets of Sector 34, a dedicated measurement was put in place to measure the magnet splice resistance. These measurements were compared with the quench test data and it was found that the error could easily be 20 nΩ (nominal about 0.3 nΩ), similar to the error in the calorimetric measurements made in LHC.

AVAILABLE DATA FROM THE SM18 MAGNET TESTS

Hall SM18 is the test facility used during the 4 years campaign dedicated to LHC magnets reception and validation.

The data recorded during the magnet acceptance tests primarily contained the electrical insulation of the coils and heaters and at the maximum field strength [3]. A second objective was to determine the field quality through magnetic measurements. No specific splice resistance test was performed.

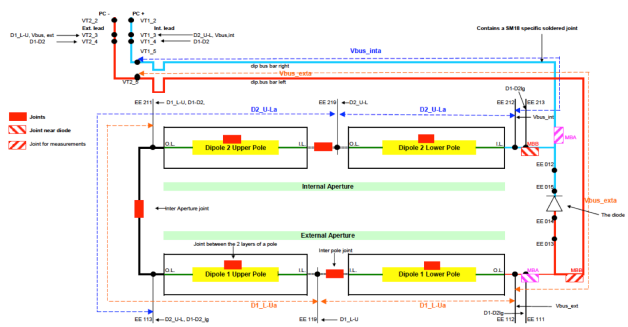


Figure 1: Electrical connections and splices of a LHC dipole.

Nevertheless the quench recorder system, used during these qualification tests, allowed acquiring and archiving hundreds of voltage signals, before and after the quench. These measurements were taken over several sections of the magnet circuit, through the existing voltage taps (see Fig. 1) inside and around the dipole. It is not possible to directly measure the voltage at a specific splice due to the positioning of the voltage taps. The data only allows for the comparison of voltage values in different segments of the magnet for constant currents: any abnormally high value in the difference of measured voltages is an indication of a relatively large resistance in the splices. The method is applicable in the assumption that the probability of having two or more large resistances in the same dipole, whose effects cancel each other perfectly, is very low.

We list here (see Fig. 2) the voltage differences used in order to infer the quality of the two inter-poles, the inter-aperture and the diode splice resistances.

$$V_{D1\_L-Ua} = V_{D1\_L-U} - V_{bus\_Ext}$$
$$V_{D2\_U-La} = V_{D2\_U-L} - V_{bus\_Int}$$
$$V_{D1-D2a} = V_{D1-D2} - V_{bus\_Ext} + V_{supra\_Ext\_lead} - V_{clamp\_Ext} + V_{bus\_int} - V_{supra\_Int\_lead} + V_{clamp\_Int}$$
$$V_{bus\_Inta} = V_{bus\_Int} - V_{supra\_Int\_lead}$$
$$V_{bus\_Exta} = V_{bus\_Ext} - V_{supra\_Ext\_lead}$$
$$V_{Diodesplice} = \begin{cases} V_{bus\_Exta} & \text{for MBA dipoles} \\ V_{bus\_Inta} & \text{for MBB dipoles} \end{cases}$$

Figure 2: Voltages used for the splice analysis.

Data Selection

The most suitable tests for our analysis were those with a long constant current at different current levels. These are present in the quench heater test at 1.5, 3, 12 and 13 kA. They have sometimes a very short constant current part, due to the fact that the operator triggered the recording manually when the current reached the desired value. The constant current recordings vary between 2 and 15 seconds at a sampling rate of 5 kHz (see Fig. 3).

The data taken at 12 and 13 kA have a recording at constant current before and after the trigger, but due to the uncertainty of the amplitude of the offset in the voltage signals, one needs a reference at lower or zero current for comparison.

Constraints of the Measurement Electronics

The electronics used for the amplification of the voltage tap signals consists of two parts connected in series: an isolation amplifier and a variable gain operational amplifier.

This last one can amplify or attenuate the signal depending on the gain setting that is adjustable with a rotary switch. The gain value for each channel has to be manually entered into a configuration database as there is no electronic read-out.

It also has a potentiometer to adjust the zero offset, which is done manually at the start of each measurement campaign.

These manual operations generate some risks, in terms of exactitude of the database content but also for the measured signals.

ANALYSIS METHODS AND TOOLS

High Frequency Data

Our first method determined the resistance of the splices, shown in Figure 3, using the voltage signals over the measured magnet current. For each magnet analysed, these values have been read from high current (around 12 kA) and low current (3 and 1.5 kA) data. Then formulas were applied to return only differential voltages of inter-pole, inter-aperture, internal bus and external bus splices. Afterwards a best-fit interpolation was done on the curves U = f (I) for the selected signals to obtain splice resistance of the related connections. We will refer to this as the fit method. It assumed that the offset is stable in time and is thus very sensitive to any offset drift. As this drift was a concern, a second method has been applied to improve the accuracy of the results.

Most of the magnets have their measurement sequence executed over several days. This leads to potential sources of offset mainly due to electronics drift over time and SM18 hall temperature variations.

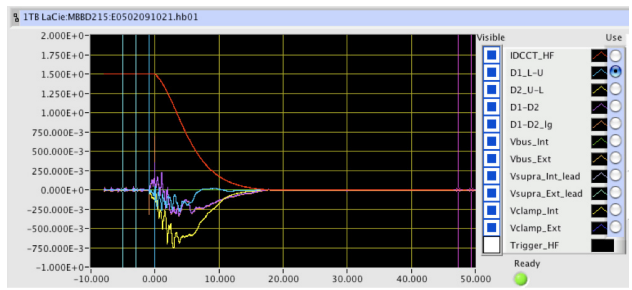


Figure 3: Magnet current and voltage signals from a 1.5 kA SM18 data file.

We assumed that offsets were constant on the voltage signals in the short time span before and after the quench, we have used the SM18 data to compensate the offsets. The resistance was then calculated in a differential way, subtracting these two voltages over the current before the quench.

The limitation of this method was that only the 1.5 kA measurements were usable, because the data taken at higher currents did not include the needed stabilized signals. Even for some quenches at 1.5 kA, signals did not always stabilize well, due to the dynamic behavior of the magnetic field and could not be used.

Another problem we found was that in about 10% of the measurements the gain for the signals Vbus_Int and Vbus_Ext was systematically too high by a factor of 400. By reading the operation logbook, it has been possible to correct these measurements. This eliminated several otherwise suspect magnets.

Low Frequency Data

After the success of the offset compensation method applied to measurements at 1.5 kA, we tried to see whether a similar method could be applied to high current data based on the low frequency signal sampling. In a short interval around the quench, the sampling frequency is 1 kHz. Outside of this interval signals are recorded when their values change by a certain minimum amount, else only once in 10 minutes, resulting in very few points. Unlike at 5 kHz sampling, offsets are compensated by the measurement electronics used for the low frequency sampling. We tried to see whether these data could be used for offset correction of the data at 12 kA. Unfortunately this was not possible, as there were too few data points in the interval following the quench.

Noise Characteristics

The voltage signals studied typically have a 1 mV noise amplitude peak-to-peak, which is much more than the DC values relevant for 10-100 nΩ resistances.

Early in the analysis we looked at frequency spectra of some signals to check if systematic features in the noise could influence the DC measurements. This turned out to be not the case. Typically there was a strong 400 Hz (and harmonics) contribution in the spectra, explained by the regulating frequency of the power converters, but no noise which could influence the DC value derived from averaging the signals measured at 5 kHz sampling frequency.

Analysis Tools

To cope with the large number of files to be processed (for 1530 measured magnets we had 23000 files) we have developed a tool for automatic data extraction and analysis.

It was designed as follows: A configuration file stored the list of the data types to use for the analysis. The tool automatically opened the files, determined the stable current plateaus and extracted the relevant voltages. Then it performed a linear fit through all the points to evaluate the resistance from the slope. Distribution plots were displayed by the application for each calculated voltage, which gave an overview of the resistances for a complete sector.

RESULTS

The Data Sample

Each dipole magnet tested in SM18 has been measured under several conditions to study its performance. Nevertheless, by only measuring those magnets that contained a long enough stable plateau in current and voltage could provide results that can be used for the resistance calculation. In addition, the linear fit method was only applied to measurements taken within a few hours of each other to avoid introducing errors due to change of conditions (drift in electronics, temperature...).

The measurement system has a 16-bit accuracy over ±10V, which gives 0.3 mV of precision. The amplifier gain was x5 on the D1 and D2 channels. The analysis used current data from 1.5 kA to 12 kA. From this we could calculate that 1 bit corresponded to 5 nΩ at 12 kA. Due to averaging over several hundred points the resolution was improved to 0.5 nΩ.

The distribution showed that the FWHM of the inter-pole resistance was ~10 nΩ. The RMS of the inter-pole resistance per magnet was 0.5 nΩ and 1 nΩ, as expected from the resolution, however this does not take into account systematic effects such as offset drift.

In the course of the analysis we found that the data set corresponding to magnets belonging to sector 78 gave particularly bad results: 33 of the 154 magnets could not be qualified at all. In depth investigation of the data showed that in most cases, even if a current plateau had been reached it was not long enough to allow all voltage signals to stabilize. As sector 78 was filled with the first magnets, we assume that the measurement procedures and tools in SM18 were not yet well tuned and that therefore the measurement conditions were not easily comparable with the other sectors.

The complete available data set was used to crosscheck the quality of the analysis tool, in particular when the individual bench behaviors were analysed. But to get results quickly, we only performed a complete study of the resistances for those magnets from sectors that had not been measured with more accurate methods, i.e. sectors 23, 34, and 45. However, to verify the reliability of our results we performed a detailed analysis of sector 67, which had more precise QPS measurements.

Analysis of the Bench Stability

The SM18 magnet test facility consists of 12 benches arranged in 6 clusters of two, sharing the same racks of DAQ electronics [4]. After analysing the data using the linear fit method, we noticed that the amount of magnets with suspect values on one of their inter-pole splice was strongly dependent on the bench it was measured on and on the date of the measurement. We therefore performed a study in order to characterize the benches (and/or their electronics).

For this we focused on two resistances, R11 and R14, which entirely belonged to the bench, as well as the clamps connecting the dipoles to the bench. We observed that:

- Data taken for the same magnet during different periods of time or on different benches could not be combined.
- The characteristics of two benches over time were unstable.

In the course of this analysis we detected several measurements indicating bench resistances substantially higher than 100 nΩ: besides being unusable for the purpose of the determination of the dipole resistances, these measurements pointed to an error in the electronics, as they could not be physically true. Indeed, in several occasions, the gain factor was wrongly entered into the configuration files associated to the data.

Comparison of Offsets

A total of 152 usable measurements at 1.5 kA were available for the magnets that showed a high resistance in the fit method. Many of the 1.5 kA measurements made on two benches gave high values for D2_U-L.

Excluding these two “bad” benches in the offset compensation method left only a few magnets with R(D2)> 25 nΩ .

A crosscheck of the best-fit and offset compensation methods has been done using all magnets measured on one good bench. These measurements showed particularly good signals. The result was that the offset method worked very well once the gain correction was applied.

Analysis Summary

Overall there were five magnets flagged for high resistance in one or several of the D1, D2, or Diode splices. One of the five magnets was known to have a bad D2 splice; the results of the other magnets were explained by non-stabilized signals. Without gain correction, four more magnets with (very) high resistance would have been flagged.

The measurements of D2 on the two "bad" benches showed signals that were not stabilized before the end of the recording. Thus we excluded these two benches for analysis of D2 with the offset compensation method.

The number of suspicious magnets resulting from the combination of both methods turned out small enough for detailed further manual analysis.

In addition, sixteen magnets could not be properly evaluated due to the poor quality or missing data.

Another faulty magnet, which was found with QPS measurements, could not be identified unambiguously with the SM18 data, as the problematic splice was the inter-aperture one.

No other magnets were identified as having a splice resistance higher than 25 nΩ on the inter-pole or diode splices.

CONCLUSION

During this data-mining campaign aimed at finding bad magnet splices, more than 23000 magnet performance measurements were scrutinized. This has been made possible by developing a specific analysis tool, based on an existing LabVIEW™ data viewer. Adding automated pattern and signal extraction and writing analytical algorithms permitted to get an overview of the splice resistance of four LHC sectors. From this, five magnets having high internal splice resistance have been flagged.

REFERENCES

[1] S. Claudet, G. Kirby, A. Perrin, K. Dahlerup-Petersen, G. de Rijks, A. Siemko, A. Verweij and R. Wolf, "Procedure for Repowering Sector 5-6, 6-7, 7-8 and 8-1 to Assess the Quality of the Electrical Joints of the Main Dipole Circuits", CERN, Geneva, November 2008, EDMS Document N.977374 (2008).

[2] C. Charrondiere et al., "Dipole Magnet Splice Resistance from SM18 Data", CERN, Geneva, LHC Project Note 430, April 2010 (2010).

[3] J. Billan, L. Bottura, M. Buzio, G. D'Angelo, G. Deferne, O. Dunkel, P. Legrand, A. Rijllart, A. Siemko, P. Sievers, S. Schloss, L. Walckiers, "Twin rotating coils for cold magnetic measurements of 15 m long LHC dipoles", IEEE Trans. Appl. Supercond., Vol. 10, Mar. 2000, pp. 1422-1426.

[4] A. Raimondo, P. Coutinho-Ferreira, G. D'Angelo, H. Franca-Santos. M. Gateau, M. Peryt, H. Reymond, A. Rijllart, A. Siemko, "The Hardware Recognition System for the SM18 LHC Cryomagnet Test Benches", in *Proceedings of* ICALEPCS'2003, Gyeongju, Korea, October 2003.

THE CERN ACCELERATOR MEASUREMENT DATABASE: ON THE ROAD TO FEDERATION

C. Roderick, R. Billen, M. Gourber-Pace, N. Hoibian, M. Peryt, CERN, Geneva, Switzerland

Abstract

The Measurement database, acting as short-term central persistence and front-end of the CERN accelerator Logging Service, receives billions of time-series data per day for 200,000+ signals. A variety of data acquisition systems on hundreds of front-end computers publish source data that eventually end up being logged in the Measurement database.

As part of a federated approach to data management, information about source devices are defined in a Configuration database, whilst the signals to be logged are defined in the Measurement database.

A mapping, which is often complex and subject to change/extension, is required in order to subscribe to the source devices, and write the published data to the corresponding named signals.

Since 2005, this mapping was done by means of dozens of XML files, which were manually maintained by multiple persons, resulting in a configuration that was error prone.

In 2010 this configuration was fully centralized in the Measurement database itself, reducing significantly the complexity and the actors in the process. Furthermore, logging processes immediately pick up modified configurations via JMS based notifications sent directly from the database.

This paper will describe the architecture and the benefits of current implementation, as well as the next steps on the road to a fully federated solution.

MEASUREMENT SERVICE OVERVIEW

The Measurement Service (MS) [1] is a vital component of the mission critical CERN accelerator Logging Service [2].

As shown in Fig. 1, the MS processes are responsible for subscribing to data published from accelerator equipment and persisting the results in either the Oracle Measurement database (MDB) or in SDDS (Self Describing Data Sets) files.

An optimized Java API has been developed, by which data for some 200 thousand signals are persisted to the MDB, from where a sub-set of the data is later transferred to the Oracle Logging database (LDB) for long-term storage. A complimentary logging to SDDS files is used typically to store data for complex data structures such as image captures or multi-megabyte beam profiles.

Java APIs are provided for extracting data from both the Logging Service databases and the SDDS file repository.

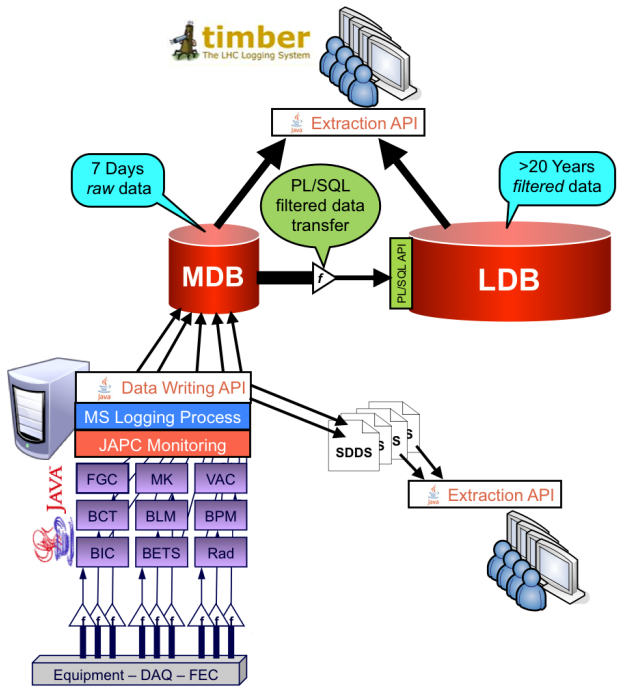


Figure 1: Measurement Service architecture and position within the Logging Service.

CONFIGURATION ESSENTIALS

In order to log data, a number of configuration steps are required:

Data Logging

It is a prerequisite to register all signals for which data needs to be logged in the MDB. The names of the signals must adhere to the well-established naming conventions, and complimentary information such as units and a meaningful description are required.

For signals whose data should be kept beyond the 7-days supported by the MDB, the filtering criteria (dead band and dead time smoothing, precision etc.) for transfer to the LDB must also be defined.

Data Acquisition

In order to acquire values to be logged, controls middleware (CMW) subscriptions via JAPC [3] need to be configured to obtain publish values. This implies declaring which device-properties to subscribe to, under which conditions (i.e. only for beams to SPS, LHC, etc.), and how the data should be time stamped (either the acquisition time, or the start time of the corresponding magnetic cycle). In addition, it is possible to specify special conversion code to be applied to published values prior to logging.

Bridging the Gap

Once signals are defined and JAPC subscriptions configured, it is necessary to bridge the gap between the two domains. This requires that individual fields of device properties are mapped to logging signals. For data published in array format, it is possible to configure the mapping of either individual elements or a subset of elements to scalar or array type logging signals respectively.

INITIAL IMPLEMENTATION

The initial implementation of the configuration of the MS was comprised of 3 components:

- Logging signals pre-registered in the MDB, and defined by means of data providers completing an Excel template with the details of the signals to be logged, which was validated by members of the logging team responsible for the MDB.
- JAPC parameter subscriptions defined in an XML file, filled by data providers, and validated by members of the logging team responsible for the MS processes.
- JAPC to logging signal mappings defined in an XML file, also filled by data providers, and validated by members of the logging team responsible for the MS processes.

This approach was quite heavy for the data providers, and susceptible to errors due to the need to manually ensure the consistency across the three separate configuration components. Flexibility was also lacking, since adding a new logging signal or modifying an existing one implied modifying 2 XML files, committing them in the code repository, and re-releasing and re-starting the relevant MS process. Although this approach was used successfully during 5 years, there was definitely scope to improve.

A FEDERATED APPROACH

The CERN accelerator control system is fully data-driven, based on several distributed Oracle databases, which collectively cover all data domains such as layout, asset management, controls configuration, and operational data. As part of a federated approach to data management, each database is considered the source for data in a particular domain, and data synchronization procedures are in place – either executed automatically or manually – to propagate necessary data to dependent databases [4].

The Controls Configuration database (CCDB) is the source for all data that describes the configuration of the control system. This includes amongst many other things the definition of multiple device-property models that describe deployed devices, and their available properties and fields, which can be subscribed to and/or modified [5]. The device-property data can be quite dynamic, mainly due to evolving requirements or to follow hardware or software developments.

The Goal

To achieve the goal of ensuring a smooth uninterrupted running of the MS, it is essential to have a consistent measurement configuration that is fully synchronized with the CCDB source data. That is to say that any changes to CCDB device-property data (such as device renames, or changes of properties and fields) should automatically be crosschecked against the MS configuration. Compatible changes should be immediately propagated to the active MS configuration, while non-compatible changes (such as removal of a property or field) should generate an alert.

On the Road

In order to reach the aforementioned goal, the first step was to provide an infrastructure inside the MDB to define the *complete* MS configuration metadata using a relational model. This metadata needed to include not only the existing definitions of the signals to be logged, but also the JAPC subscription parameters (device-property), and the mapping between the domains, as mentioned above.

Such an approach was deployed during 2010, making it possible to simplify the code and configuration of the many MS data logging processes, as well as bringing other advantages:

- A MS data logging process can now simply retrieve its full configuration by calling a single method with its process name as input.
- Requestors for data logging only need to make a single request to configure logging with a common Excel template for defining all parameters. This eliminates the possibility of having inconsistencies in the metadata describing JAPC subscriptions, logging signals, and mappings, both at the time of initial definition, and for subsequent updates to the device-property model data.
- The time required to configure logging of additional signals, or modify existing configurations is reduced significantly.
- With a single point of definition for MS process configuration, using an Oracle relational database, it is possible to envisage the simple propagation of CCDB device-property model data to the MDB using custom PL/SQL procedures.

ONLINE UPDATES

Previously, whenever requests to modify MS process configurations had been fully treated, it was necessary to restart the process concerned in order to apply the changes. This implies stopping all existing JAPC parameter subscriptions, and then restarting them based on the latest configuration. Considering that on average each MS process treats data for tens of thousands of signals, this approach was both time consuming and implied undesirable data loss. For example, to add the logging of a single new signal, it would be necessary to restart the related MS process and incur a non-negligible

downtime of logging of many of the other existing signals already being logged.

To minimize such data loss and make configuration updates more flexible, an online update mechanism was put in place (see Fig. 2):

- A virtual device-property was defined in the CCDB to indicate if a MS process configuration has been modified.
- The MS process code was adapted to subscribe to this property at start-up via JAPC.
- Once a MS process configuration is modified in the MDB, an updated value for the aforementioned virtual device-property is published directly from the MDB using Oracle Advance Queuing [6].
- When a MS process receives a notification that its configuration has been modified, it retrieves the new configuration from the MDB, compares with its current configuration, and then makes the minimum number of JAPC re-subscriptions necessary in order to get an up-to-date process configuration running.

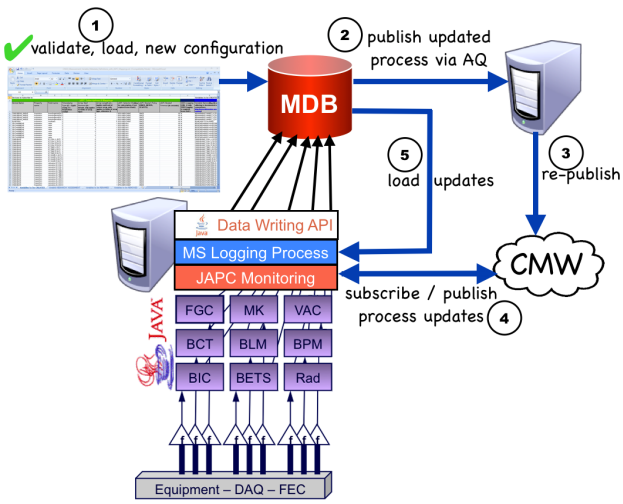


Figure 2: Online updates of MS configuration.

In summary, modifications to MS configurations can now be made online, and no longer imply data loss for other signals.

RESOURCE OPTIMIZATION

The MS writes up to 5.4 billion records per day (i.e. ~270GB) to the MDB. In order to optimize resources, some additional data driven features have been integrated into the MDB data writing API:

Log-on-Change

Depending on device-property structures and / or other requirements (e.g. continuous content for Fixed Displays), it is quite common for unchanging values to be published at relatively high frequencies. Examples include status flags, temperatures, counters etc. In most cases it is sufficient to log these values only when they change.

To implement this functionality, additional attributes were defined in the MDB metadata for each signal:

- A flag to turn on/off the on-change comparison.
- An integer value to specify a precision for on-change comparison.
- A flag to qualify if the precision refers to a number of digits left or right of the decimal point, or to a number of significant figures.
- An interval, after which to force logging of a published value even if it is not considered as changed.

These new attributes were exposed in the MDB data loading API, which then caches the last logged value for each signal which has on-change logging enabled. The new attributes are used to establish if new values can be considered to have changed sufficiently with respect to the last logged value. Newly published values are then only published on-change, or if the specified interval since the last logged value has elapsed.

Value Rounding

Numeric data is stored in the MDB using the Oracle NUMBER data type, which requires a variable number of bytes according to the scale and precision of the numeric values to be stored.

Many device-property field values are published with a much higher precision than the real physical accuracy (e.g. due to calculations prior to publishing), or what is actually required. An example would be a thermometer voltage-to-temperature conversion with 10 digits of precision.

In the same manner as for the on-change logging described above, additional metadata attributes were defined and exposed per signal – allowing the MDB data loading API to round values before writing to the MDB, thus saving storage, and improving I/O response times.

Not Just Resources

It is worth noting that while the described optimizations reduce the amounts of network activity, processing to load data, and storage required, the main benefit is actually for the users of the data, since having only significant data logged improves retrieval times and facilitates data analysis.

A CLEAR VIEW

With the configuration of MS processes and MDB signals now fully defined within the MDB, it was necessary to provide data providers, consumers, and MS experts with a means to visualize this configuration data.

To satisfy this requirement, a Web interface was developed using Oracle Application Express (APEX) to search and report on the current MS configuration (see Fig. 3).

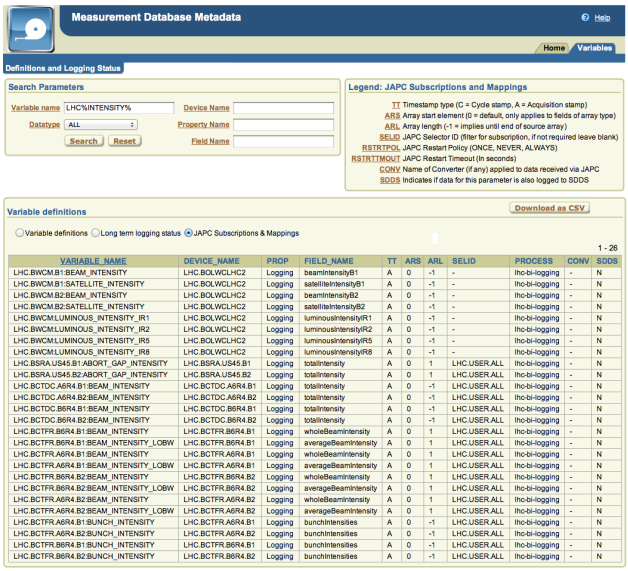


Figure 3: Measurement Service configuration report.

The report currently includes data acquisition configuration per MS process, applied data filtering criteria, and long-term storage status. In addition, the interface allows data providers to download existing configurations to a standard Excel template, from where configurations can be modified and/or extended and then submitted to service administrators for validation and integration into the system.

NEXT STEPS

In order to continue on the road to a fully federated data management solution, the links to the CCDB device-property model data need to be enforced, with automatic checks, updates, and notifications in place as outlined above.

The interfaces for browsing MS configuration data, and CCDB device-property data should be linked in both directions. This will simplify the process of browsing and configuring the MS for data consumers and providers alike. For example, it will be possible when browsing the device-property data to see which property fields are logged, and when browsing the MS configuration, to see the underlying data types of the device-property fields, or additional fields which may need to be logged.

Flexible Data Extraction

The Logging Service data extraction API currently only supports extraction of data based on signal names. However, it is foreseen to reuse the MS configuration data that maps JAPC parameters to logging signals within the data extraction API, thus allowing the extraction of logged data based on JAPC parameters. This will give users of logged data a more flexible means to retrieve data and facilitate the development of certain types of applications such as fixed displays that can optionally display a historical set of data.

SUMMARY

With respect to the initial implementation, the current infrastructure has already significantly simplified and improved the robustness of the process of configuring the Measurement Service.

The Measurement Service is well on the road to federation in the distributed database environment of accelerator controls. The foundations have been established, with all configuration data now centralized in a relational Oracle database, and the next steps to achieve the desired goals are foreseen.

REFERENCES

[1] M. Gourber-Pace et al., "Status Report of The Measurement Service for the CERN Accelerator Logging", ICALEPCS'09, Kobe, Japan, October 2009, TUP106.

[2] C. Roderick and R. Billen, "Capturing, Storing and Using Time-Series Data for the World's Largest Scientific Instrument", November 2006, CERN-AB-Note-2006-046 (CO).

[3] V. Baggiolini et al., "JAPC - the Java API for Parameter Control", ICALEPCS'05, Geneva, Switzerland, October 2005, TH1_5-8O.

[4] R. Billen et al., "Accelerator Data Foundation: How It All Fits Together", ICALEPCS'09, Kobe, Japan, October 2009, TUB001.

[5] Z. Zaharieva et al., "Database foundation for the Configuration Management of the CERN Accelerator Controls Systems", ICALEPCS'11, Grenoble, France, October 2011, MOMAU004.

[6] K. Kostro et al., "On-change Publishing of Database Resident Control System Data", ICALEPCS'09, Kobe, Japan, October 2009, TUP013.

DATABASE AND INTERFACE MODIFICATIONS: CHANGE MANAGEMENT WITHOUT AFFECTING THE CLIENTS

M. Peryt, R. Billen, M. Martin Marquez, Z. Zaharieva, CERN, Geneva, Switzerland

Abstract

The first Oracle®-based Controls Configuration Database (CCDB) was developed in 1986, by which the controls system of CERN's Proton Synchrotron became data-driven. Since then, this mission-critical system has evolved tremendously going through several generational changes in terms of the increasing complexity of the control system, software technologies and data models. Today, the CCDB covers the whole CERN accelerator complex and satisfies a much wider range of functional requirements. Despite its online usage, everyday operations of the machines must not be disrupted.

This paper describes our approach with respect to dealing with change while ensuring continuity. How do we manage the database schema changes? How do we take advantage of the latest web deployed application development frameworks without alienating the users? How do we minimize impact on the dependent systems connected to databases through various APIs? In this paper we will provide our answers to these questions, and to many more.

INTRODUCTION

The Controls Configuration Database (CCDB) provides the Configuration Management services for the Control System of all CERN accelerators [1]. It is a multifaceted software infrastructure composed of many interrelated components at the heart of which lies an instance of Oracle® database. CCDB relies on web deployed tools for data browsing and editing, and is accessible through Application Programming Interfaces (APIs) written in different programming languages. It is a component of a distributed database environment for the CERN accelerator complex and as such, it is linked to several other Oracle instances.

The CCDB forms the data foundation of the accelerator Control System and is used online for all controls operations. Consequently, full availability and reliability needs to be guaranteed to ensure accelerator operations and thus the CERN physics program.

No Downtime Allowed

Fortunately, the occurrences for which the CCDB was the cause of accelerator downtime have been extremely rare. However, in order to maintain its high quality, the CCDB has to evolve over time, similarly as all equipment that needs to be maintained and upgraded.

The accelerator operations follow a precise schedule per accelerator with technical stops with different frequencies and varying lengths. As for any hardware or software intervention that affects machine operations, deployments of CCDB changes need to be carefully planned in order to minimize the risk of disruptions during physics exploitation. Typically, we are granted an hour of database unavailability, every two months, to be scheduled on a single target day. Interventions requiring a longer downtime have to be scheduled during shutdown periods, i.e. outside the physics program.

Preparing the Interventions

Due to the severe constraints on deployment time and full proof quality of the intended modifications, every change undergoes prior multistage testing and comes bundled with a clear procedure for rolling back to the previous state. The sections that follow go into details of how these changes in the CCDB are dealt with. In order to better understand what type of changes are concerned, it is useful to recall the major milestones in the Controls Configuration Database.

CCDB HISTORY

The list below sketches the major modifications that were introduced in the CCDB over its 25 years of existence [2]. The important database schema modifications, interfacing technologies and functionality leaps are indicated.

- 1980 – Creating a centralized file-based data storage for some components of the Controls System
- 1986 – Introduction of Oracle RDBMS: CCDB birth
- 1987 – Data extraction scripts (embedded SQL in Pro*Fortran, re-implemented in Pro*C later on)
- 1995 – User interfaces based on Oracle Forms and PL/SQL Web Toolkit (OWA)
- 1999 – Java Directory Services
- 2003 – Introducing FESA Device-Property model
  2004 – Migration of data browsing interface to Oracle APEX technology
- 2005 – Introducing authentication and authorization mechanisms
- 2005 – Big-bang refactoring of the database schema
- 2005 – Introducing the Session Auditing and History Recording Framework
- 2006 – Redesign of the data editing interfaces using Oracle ADF technology
- 2007 – Introducing Hardware device-property model
- 2010 – Introducing Virtual device-property model
- 2011 – Introducing Configuration Change Management and Status Accounting in the CCDB [1]

The original database in 1986 was based on Oracle version 5, which was migrated through each major version up to the current Oracle 10*g*. The upgrade to

Oracle 11*g* is scheduled in January 2012 during the next winter shutdown.

The CCDB is now under the responsibility of the second generation of database engineers. Over the 25 years of the lifetime of the CCDB, programming languages and technologies change, applications and software come and go, but the data remains.

REASONS FOR CHANGE

A look at the timeline above makes it clear that the driving forces for change can be classified into the following categories:

- Changing user requirements.
- Changing software technologies.
- Internal refactoring to improve software quality and reduce maintenance burden.

Changing User Requirements

Changes in the user requirements constitute the principal reason for CCDB modifications. New functionalities are requested, existing ones change, obsolete ones are dropped, new components of the Control System are in need of data driven configuration, and users' expectations towards the IT systems evolve. Although most frequent, these modifications benefit from the highest rate of acceptance because their rationale is well understood by the users.

Changing Software Technologies

The second category of changes is forced by the evolution of the technology stack that CCDB depends upon. Since the early days of CCDB we have been using Oracle products and now that Java is also part of Oracle we are nearly 100% dependent on this vendor's products. The potential vendor lock-in is not the main concern, but as Oracle technologies evolves so needs the CCDB.

Database upgrades are necessary to ensure the long-term continuity and support, and to take advantage of new features, functionality and efficiency. These upgrades are never carefree, but have been consistently programmed towards the stable, terminal release of a major version.

The data-driven client APIs have evolved from Pro*C precompiled code to Java (although some Pro*C legacy has not yet been phased out).

For interactive user interfaces, Oracle Forms have been upgraded from version 4.5 up to 9*i* and finally replaced with J2EE-compliant Oracle Application Development Framework (ADF).

Web-deployed interfaces and reports, originally generated by the PL/SQL OWA module, left its place to Oracle HTMLDB which evolved into Oracle Application Express (APEX).

All these technological changes are imposed upon the developers as well as upon the end-users. Often this requires a change in the users' habits and requires well devised communication campaigns and careful deployment. The human factor must not be neglected here as no software product can be successful unless its users are happy.

Software Refactoring

The last – but not least – driver for change comes from our own quality assurance standards. We are aware that in order to maintain the high quality of the systems we provide, there is a need to regularly revisit existing production code and evaluate following considerations:

- Improving the performance of interfaces.
- Streamlining data propagation.
- Simplifying workflows.
- Improving data quality and integrity.
- Getting the most out of new technologies.
- Complying with our own coding and data management standards.

Modifications that result from these considerations do not necessarily give any added value or visible impact for the end user, but is purely for reasons of code maintenance and efficiency of the development team. In addition, these changes are not always transparent to the end user. For example, the introduction of referential integrity constraints across the database – which were not present in the original design – resulted in error messages seen by the users. This modification greatly enhanced the quality of data, as opposed to recording incoherent, erroneous information prior to the refactoring.

It has to be noted that all three categories of changes have an impact on the users of the services provided by the database. These users are not only human actors, but also the dependent computer systems that are linked to the CCDB. The following sections outline our time-tested strategies for managing this impact.

STABLE INTERFACES

By having well defined interfaces towards the external systems, the CCDB can afford to change underlying implementation quite freely and transparently. Data is exposed through database views. Unless a very profound refactoring takes place, the structure of views can remain identical even if underlying tables are altered. Backwards compatibility after data model changes is a primary concern. If this cannot be ensured, the interfaces have to be renegotiated with the users, and the whole deployment process becomes much more complicated. For example, hundreds of front end computers may need to be rebooted to force the correct runtime deployment. Depending on the functionality of the front end, this is something that can only be scheduled during a technical stop.

We have separate dedicated database accounts with different sets of privileges corresponding to different usages of data. This way we control in a very precise manner the data access – for reading or for editing – by a particular set of users. For distinct clients, specific views are provided, which all point to the same underlying data.

An additional layer of isolation is provided by the APIs that are exposed to the users. Nowadays, these are written in Java or PL/SQL with XML being the representation of

choice for data transfer. These APIs have proven to be very stable . One example is Java Directory Services [3] which dates back to 1999, but has been recently refactored to improve the performance and benefit from the many useful features provided by the Spring framework [4]. Even if its internals changed to a very large extent, and some new interfaces were added, the existing ones remained fully backwards compatible.

STAGED TESTING AND DEPLOYMENT

Over the years, we have worked out a staged environment that provides a complete and efficient framework for testing and deployment. We have provided dedicated instances of the CCDB in four distinct environments: DEV, TEST, NEXT, and PRO as shown in Fig. 1.

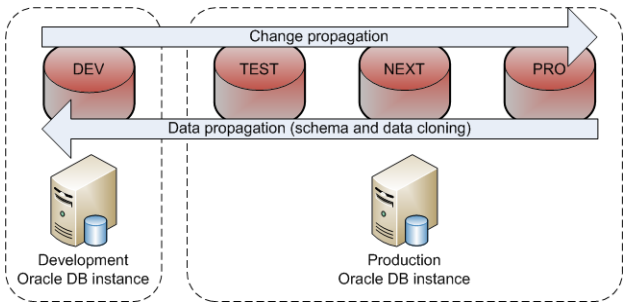


Figure 1: Overview of the four CCDB environments.

DEV is the development environment in which the database schema is installed at a development database instance. It is our "try-it-out" environment where no stability is required. DEV is used for proofs of concepts, general development and introducing new code.

TEST is the environment for functional testing. Tests are started with the database being a clean 1:1 copy of the production schema. Consequently, new code from DEV is copied across and all necessary tests are carried out in isolation from other systems that depend on CCDB. The following tests are against all existing applications that are affected by the newly introduced changes.

NEXT is a relatively recent environment (created in 2008) that has been introduced for integration testing. It is a part of the Controls Testbed since 2010 [5], a fully functional vertical slice of the Accelerator Controls System. As such it has every component from the whole control system present. The NEXT environment represents the CCDB as it will be after the following deployment in PRO.

PRO is the production environment used by the Controls System. It is guaranteed to be stable, as no changes are performed outside of scheduled interventions.

TEST, NEXT and PRO environments are hosted on the same database server, which allows us to test the performance and scalability in real life conditions. For resource demanding testing against TEST and NEXT, precautions are taken in order to avoid performance degradation of the production environment.

INCREMENTAL APPROACH

Historically some major change sets were applied to CCDB as "big-bangs" but with the LHC in full operation, this type of revolution is excluded in order to avoid any disturbance noticeable by the user community. Therefore, deployments are rolled out adiabatically and step-wise over relatively long periods of time. This method is applied to the vast refactoring of the data management services for the CERN Front-End Software Architecture (FESA) framework [6]. The objective is to fully rationalize the database schema and to eradicate inefficient XML objects inside database in favour of a relational data model. The FESA workflow is complex and consists of controls device class modelling, deployment and instantiation phases. These workflow phases are tackled one at a time, passing through all four environments. Passing to the next one is only done when the expected behaviour is ensured. This way, an effective rollback strategy is in place in case of failure and the impact on users is limited. Moreover, even when new database schema and APIs are deployed, the previous ones are maintained in parallel for a few weeks to guarantee redundancy in case of problems. The same is valid when new user applications are put in place – the old ones are supported, if possible, for a certain period (sometimes for months) in order to ensure that the users have accepted the new functionality.

The incremental approach has also been applied for the replacement of PL/SQL OWA based CCDB Data Browser with the new one built using Oracle APEX [7]. The full functionality of the existing set of pages needed to be reproduced, while adding support for new sets of data. Navigation and overall usability was to be improved, providing a new, contemporary look and feel. The major reports were developed first – i.e. roughly 80% of all content – and deployed for public use, followed by continued adding of the remaining ones. However, to gain user acceptance, a proactive communication campain and feedback channels were put in place. In parallel, the outdated data browser was kept running for one year without any development effort. Eventually access to the obsolete service was blocked, in line with the announced planning. A very similar strategy has been followed when replacing more than 150 Oracle Forms with their Oracle ADF successors.

We are convinced that incremental approach to change is a valid one. This conviction is further strengthened by looking at some examples from the "real world", especially in the domain of Web applications. One notable example of incremental functionality enhancements is Google Mail where new features are added one at a time and there is a possibility for the users to switch them off, at least for a while.

CAREFUL PLANNING

With all CERN accelerators dependent on CCDB for their everyday operations, there is no room for

improvisation when it comes to rolling out new features. Every deployment requires very careful planning.

The impact of every change to be deployed is assessed right from the start. Deployment scripts are prepared with rollback paths at every step of the process. An exact copy of the PRO environment in NEXT is created and the timed execution of the scripts is carried, indicating the time to be allocated for the final deployment in PRO. The interventions are planned in line with the accelerator schedule in order to have zero impact on the machine operations. Announcements are prepared and sent to mailing lists of the user community.

With this type of preparation, the only action on D-day is to execute the plan. In case of any mishap, a clear exit path is available and possible at every stage. In addition, every action is recorded and stored in a secure file system in a dated folder for future reference.

This scenario ensures full transparency and traceability of each software deployment session, essential for the overall quality assurance process.

DEALING WITH HUMAN FACTOR

Dealing with the human factor is probably the most delicate part of the whole software change process in the context of CCDB. The accelerator community is large and diverse from control room operators to equipment specialists and accelerator physicists. Their expectations related to the provided tools are high, but also vary for each accelerator, due to operational habits which have not yet fully converged. As designers and developers of those tools, we have to find the balance between generic implementations and specific functionality.

Twenty-five years ago, the CCDB was deployed in the scope of the CERN Proton Synchrotron complex, without covering the larger machines. Data entry was centralized and done by a team of two dedicated data management engineers, working in close collaboration with operators and equipment experts. Since 2003, with a second generation of database engineers, the CCDB scope has been extended to cover the complete accelerator complex. Focussing on the data model, interfaces and business logic, the ownership of configuration data has been transferred to the responsible people of the hardware or software, which needs to be configured through CCDB. To this end, data entry tools have been adapted, authentication and authorization mechanisms introduced and the people trained. For reasons of traceability, every single data manipulation, executed by a user, is recorded together with time stamp and session information. This mechanism also enables the possibility of reverting to a previous data situation. Currently, more than 300 unique users are active across all applications of the CCDB.

Whenever an end user tool is to be refurbished, the key users are involved very early in the development process. They are invited to test the iterative versions of the software and to provide feedback. Prior to a deployment in PRO, the changes and their possible impact are communicated to all concerned parties via the relevant announcement channels.

CONCLUSIONS

The Controls Configuration Database is an evolutive system and as such it is subject to change. However, considering that it lies at the heart of the CERN accelerator complex, there is no room for improvisation when deploying new features. The successful strategy for dealing with change that has been put in place is based on the following guidelines:

- Involve end-users right from the start, throughout the design and development process
- Provide four separate environments for development, unit and functional testing, integration testing (TestBed), production
- Analyze the impact of a change and try to apply only backward compatible changes
- Communicate timely, clearly and transparently on scheduled intervention and their impact
- Coordinate the upgrades with impacted clients

By respecting these guidelines, we have been able to perform and manage changes and have them accepted by the user community.

REFERENCES

[1] Z. Zaharieva et al., “Database Foundation for the Configuration Management of the CERN Accelerator Control Systems”,ICALEPCS’11, Grenoble, France, Oct-2011

[2] J. Cuperus, R. Billen, M. Lelaizant, “The Configuration Database for the CERN Accelerator Control System”, Icalepcs2003, Geongeoeng, Korea

[3] J. Cuperus et al., “A Directory Service for the CERN PS/SL Java Programming Interface”, Icalepcs’99, Trieste, Italy

[4] http://www.springsource.org/

[5] J.Nguyen Xuan, V.Baggiolini, “Testbed for validating the LHC controls system core before deployment”, Icalepcs2011, Grenoble, France, Oct-2011

[6] M. Arruat et al., “FESA 3.0”, ICALEPCS’11, Grenoble, France, Oct-2011

[7] Z. Zaharieva, R. Billen, “Rapid Development of Database Interfaces with Oracle APEX, used for the Controls Systems at CERN”, ICALEPCS’09, Kobe, Japan, Oct-2009, THP108

CERN ALARMS DATA MANAGEMENT: STATE AND IMPROVEMENTS

Z. Zaharieva, M. Buttner, CERN, Geneva, Switzerland

Abstract

The CERN Alarms System - LASER is a centralized service ensuring the capturing, storing and notification of anomalies for the whole accelerator chain, including the technical infrastructure at CERN. The underlying database holds the pre-defined configuration data for the alarm definitions, for the Operators alarms consoles as well as the time-stamped, run-time alarm events, propagated through the Alarms Systems.

The article will discuss the current state of the Alarms database and recent improvements that have been introduced. It will look into the data management challenges related to the alarms configuration data that is taken from numerous sources. Specially developed Extract-Transform-Load (ETL) processes must be applied to this data in order to transform it into an appropriate format and load it into the Alarms database.

The recorded alarms events together with some additional data, necessary for providing events statistics to users, are transferred to the long-term alarms archive.

The article will cover as well the data management challenges related to the recently developed suite of data management interfaces in respect of keeping data consistency between the alarms configuration data coming from external data sources and the data modifications introduced by the end-users.

INTRODUCTION

The Alarms Service, LASER, is a critical system, used during the Operation of CERN's accelerators.

The existing service was developed for handling the abnormal situations for all accelerators and with the Large Hadron Collider in particular [1]. Previous implementations of the system were done for the PS complex [2] and for the SPS complex [3].

The alarms data management is an important part of the Alarms service. Already since the beginning of the 1980s there was an idea to implement a data driven alarms system for the PS Complex Controls.

Since 1986 a relational database and in particular the Oracle® RDBMS was chosen as the basis for the alarms data management.

SCOPE

At present the data stored in the Alarms Database covers the needs for an alarms service for all accelerators at CERN. The database contains data for the Fault States (abnormal situations) of the accelerators' devices controlled by the Controls System, e.g. magnets, beam monitoring devices, etc. The data stored in the LASER DB is used for the Technical infrastructure alarms, e.g. alarms for the areas of electricity, cooling, etc. It covers also the safety systems, the radiation monitoring system, the computer network alarms as well as alarms data for abnormal situations in the Controls software itself.

All of the above listed domains of alarms data are called providers and there are 31 in total. This accounts for more than 3,000,000 data elements stored in the LASER DB.

AREAS OF THE ALARMS DATA MANAGEMENT

There are three distinct areas of data, which are part of the Alarms data management.

Configuration Data for the Alarm Definitions

The main pieces of data that build-up the alarm definitions are the fault family, fault member within a family and the fault code of a given family. The combination of those three pieces of data uniquely identifies an alarm at CERN and represents the fault state. Additional attributes, provided by the configuration data, are the priority of the alarm, the action to be taken, etc.

The alarm definitions configuration data is extremely important due to the principle of LASER that all fault states must be defined in the LASER DB prior to their use. The LASER service will not accept and propagated events, whose definitions are not in the Alarms database.

Configuration Data for the Users Consoles

As the name suggest the data in this area is used for configuring the displays on the Operators Alarms Consoles. It includes the operators' category structures, which help the users to group the alarm events on their screens, preferences of displayed categories on the consoles, filters for the events, etc.

Run-time Alarms Data

The run-time alarms data is the time-stamped alarm events, which are coming from more than 220 sources. An alarm event is the activation, termination or update of the fault state. The LASER server processes the events data and stores it into the LASER DB. The run-time events are distributed to approximately 50 client consoles so that the users will be notified of fault states.

OVERVIEW OF THE ALARMS DATABASE ENVIRONMENT

For the purposes of the Alarms data management several db accounts are used. The main accounts are depicted in Fig. 1 as well as the ETL processes, which are used for LASER data configuration and archiving.

Alarms Configuration Data Collection Account

The Alarms Configuration Data Collection db account is used in order to collect all alarms configuration data

from the different providers. The data could be provided in the form of different db objects, e.g. views, materialized views or tables. There are 31 starting objects, which correspond to the 31 data providers. All of those objects have a standardized structure and contain the same type of configuration data. This is done in order to solve the problem of unifying the data from so many diverse providers as well as to ease the data import process.

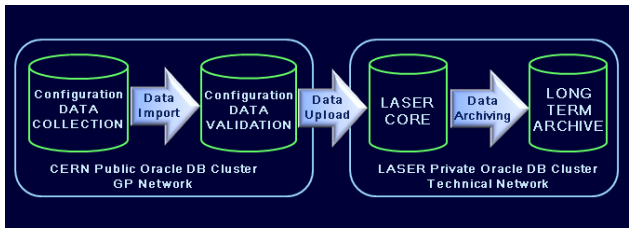


Fig. 1: Overview of the LASER data management environment – main database accounts and ETL processes.

Alarms Data Validation Account

The configuration data for the alarms definitions is not directly imported from the Data Collection into the LASER Core account. Due to the complexity of the process to validate the data and to transform it into the final destination format the loading of the alarms data passes through an intermediate stage.

LASER Core Account

The Core database account holds all the configuration alarms data, which is used directly by the LASER server. The account contains the user preferences and consoles configuration, which are extracted via a set of views and provided to the LASER consoles. The same account also stores the on-line alarm events in the form of a short term alarms archive, which contains data for the last six months. There are 200,000 time-stamped events recorded on average per day, however at peak times there could be up to 1000 events per second send to the LASER DB.

Alarms Long-term Archive Account

The long-term alarms archive has a completely denormalized schema and the structure is more like a data warehouse.

There is a PL/SQL procedure for transferring the on-line data into the long-term archive, which is executed as a db job once per day.

The long-term archive has been in operation since 2005. At the beginning of each year archiving of the previous year's data is done and between 4 to 10GB of data is added to the long-term archive. PL/SQL procedures, dedicated to this process, are used. During the long-term archiving a factor of 10 reduction of the data volume is achieved. Only the alarm record at activation time and at deactivation time for the alarm events are kept, thus all records between those two are removed.

Alarms Data Management ETL Processes

There are two main types of ETL processes, which are used for the Alarms data management.

The first type is related to the loading of the configuration data into the LASER Core account. The data needs to be validated carefully against the business rules, established for the alarms definitions and transformed into the required format. This is done in two stages due to the complexity of the data manipulations. The first process in the chain is called 'Data Import'. It takes the data from the Data Collection account and moves it to the Data Validation account. The Data Import process mainly verifies the validity of the provided data, while the second process, called 'Data Upload' has as a primary goal to transform the data into the required final format while still performing checks.

The second type of ETL processes are related to the archiving of the alarms data. There is a process that runs every day, which transfers data older than 6 months from the LASER Core to the Archiving account. Another process runs once per year in order to establish the long term archive for the previous year.

Complexity of the Database Schemas

Table 1 provides the exact figures showing the complexity of the alarms-related database schemas. The alarms database model is not a very complex one – altogether there are 207 tables and 241 constraints throughout the four main accounts. On the other hand due to the numerous validations, checks and transfers of data, part of the intensive ETL processes, there is a big PL/SQL code base.

Table 1: Alarms Database Accounts Statistics

Tables	207
Constraints	241
Lines PL/SQL code	32,000
Volume	195 GB

SERVICE QUALITY ASSURANCE

Database High Availability

One of the challenges in front of LASER DB is to provide high availability service - 24 hours a day, 365 days a year, as the Accelerators complex at CERN cannot function reliably without the Alarms system.

The high availability of the database is achieved through the use of Oracle cluster technology (RAC) which guarantees not only the hardware but also database software redundancy. The solution ensures no down time even during routine software patching of the database.

Another point in the strategy for high availability is to have a special db cluster dedicated only to LASER (private). This approach guarantees that no problem could be encountered due to other db clients on the same cluster.

Data Security

Data security is of utmost importance in the domain of alarms data management. A major improvement in this domain since 2009 was the deployment in the LASER Core account of several custom-developed diagnostics frameworks, allowing monitoring and traceability of *who* changed *what* and *when*. This is extremely important especially in the light of providing new applications to the end-users (accelerators operators) to be able to change some of the configuration data themselves.

The first framework audits every session opened in the LASER DB, while the second framework, the History one, records all data modifications and keeps them on-line, available to be accessed through a web-deployed LASER History Browser application.

Development and Testing Environments

The Alarms service is a critical one for the Operation of the Accelerators. The database part of the Alarms service is equally important and every piece of new code should be thoroughly tested.

Three environments have been provided, in order to ensure bug-free alarms data management. The ***Development*** environment is used for developing database functionality as well as developing the related Data Editing and Data Browsing applications and database APIs. The ***Test*** environment (created in 2010) is used for functional testing within the database tier as well as an integration testing environment of the vertical slice of the Alarms service – database tier and LASER server. The ***Production*** environment is the one used for the on-line Alarms data management.

ALARMS DATA MANAGEMENT CHALLENGES

Multiple Data Providers - Eliminating Scattered Sources of Alarms Configuration Data

The data management for the Controls and Operation of CERN's accelerators is implemented as a distributed database environment [9]. Previously the alarms configuration data was gathered from a significant number of databases, e.g. the Accelerators Entities and Signals Naming DB, etc. as well as some scattered db accounts.

During the last two years efforts have been put in place to eliminate dispersed sources of alarms definitions and to streamline them through the main databases, providing data to LASER DB. As a result of those efforts several db sources have been eliminated leaving only the databases shown in Fig. 2 to deliver the alarms configuration data.

An example of this strategy is the migration of the power converter alarms configuration data into the CCDB and its integration into the Hardware Controls devices framework [4]. This solution resolved the issue of synchronization between the devices data in the CCDB and the alarm definitions for the very same devices, which were previously kept in a separate db account. At present the data provided by CCDB accounts for almost 70% of the alarms definitions configuration data imported into LASER [6].

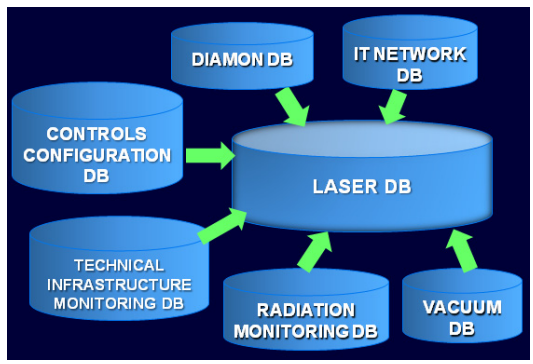


Fig. 2: Databases, providing alarms configuration data to the LASER DB.

The approach of unification of the sources of alarms configuration data guarantees not only a better maintenance of the alarms configuration data by the equipment experts, but it simplifies the process of initial import of the alarms configuration data.

Separation of Alarms Definitions Configuration Data from Users Consoles Configuration Data

An alarm system is supposed to give its users a better insight of the status of the different systems and particularly when problems arise. A badly configured alarms system (due to bad configuration data) will not achieve this goal. Typical problems, which arise in the CERN Controls Centre, are an alarm event not being seen by an operator, due to the alarm definition not being correctly configured in respect to the display category, or an alarm event appearing on the console of an operator of a given accelerator when it should not be there.

In order to solve the above mentioned issues, a clear separation needs to be achieved between the alarms definitions configuration data and the users' consoles configuration data. This means that the equipment experts should be responsible only for the alarm definitions while the operators take complete responsibility for the alarms consoles configurations data, including the assignment of alarm definitions to the operational alarms categories, which they monitor.

First steps in this direction were taken during the last year by providing some basic data editing applications through which the operators could set-up their own configurations of alarm definitions to categories assignments. One must however be extremely cautious with the data changes introduced directly into the LASER Core db as the current database model does not provide a clear split between the two alarms configuration data areas.

This problem is addressed in the new LASER Core database model, which is currently implemented in development environment and will be the basis for the renovated LASER service.

EXTENDING THE DATA MANAGEMENT FUNCTIONALITY

A lot of improvements and extensions of the alarms data management services were done during the last two years. Some of the new functionalities are related to extending the database model and some are related to providing new user applications for browsing and editing the alarms data.

Introducing Alarm Instructions

A new area of the Core LASER DB is the one of alarm instructions. This data is related to the configuration data of the alarm definitions and gives directions of what measures should be taken in case an alarm is activated.

Introducing Filtering of Alarm Events, Based on Accelerator's Beam Modes

The filtering of alarm events on the alarms consoles is an example of new functionality provided in the area of the user consoles configuration data. The alarm events that are usually displayed on the operators' consoles now could be filtered based on the beam mode of a given accelerator. This feature is extremely useful as it prevents the operators' consoles to be flooded with alarm events, which are not relevant under certain beam modes. At present the new functionality is used by the LHC Operation.

Development of a Suite of Data Management Applications

A major improvement in the Alarms Data management was the development of a suite of data management tools during the last two years. These applications give users the possibility to explore all alarms data and to maintain the configuration part of it. The choice of development technology both for the Data Browsing and for the Data Editing tools was Oracle Application Express [6].

The LASER *Data Browser* (Fig. 3) is the main reporting application. It comprise of more than 40 different reports and simple graphs with statistics, covering all areas of alarms data.

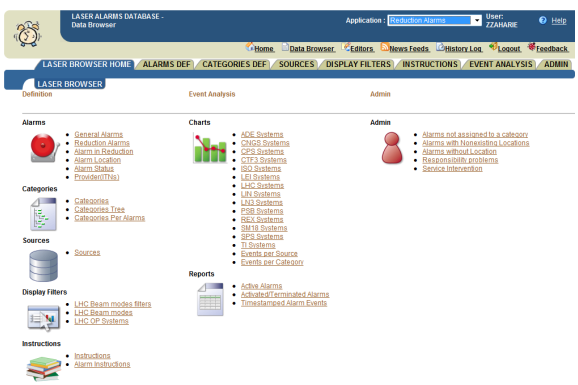


Fig. 3: LASER Data Browser.

The *Help Alarm* is a special report, which is integrated into the Data Browser. It presents all related data for a given alarm definition, which is available in the LASER DB, as well as data coming from other databases e.g. Network DB, TIMDB, DIAMONDB, CCDB, etc. The Help Alarm page is used as a reference report for any alarm by the LASER end-users.

A number of *Data Editors* were developed, targeting specific problems that the operators need to deal with. The editors are related either to giving the users the possibility to manage the Console Configurations, e.g. the *Power Converters Categories* Editor and the *TI Network Alarms Categories Editor*, or to giving the option to edit data related to the newly provided functionality in the LASER model, e.g. *Alarms Instructions Editor* and the *Beam Modes Filters Editor*. Another application, which was developed, is the *Reduction Alarms Editor*. In order to avoid an avalanche of correlated alarms on operators' consoles, special filters are declared by the users called Display or Reduction alarms. This helps to achieve a cleaner console display and to more easily spot the real cause of a problem.

CONCLUSION

The Alarms Service is a critical element, which is indispensible to the Operation of CERN's accelerators complex. The database tier plays a pivotal role and has proven to be a stable and reliable component. Continuous effort is put into its improvement through rationalization, data federation and development of new functionality at the database and interfaces level.

A new LASER Core database model is in the process of being developed catering for additional data elements for the alarms quality management and a workflow for the operators to approve the alarms configuration data.

A significant challenge in the future of the Alarms data management will be the smooth transition between the existing database model and ETL processes and the new ones to support the renovated LASER service.

REFERENCES

[1] K. Sigerud *et al.*, "First Operational Experience with LASER", ICALEPCS'05, Geneva, Switzerland, Oct-2005.

[2] J. Cuperus, "An Interactive Alarm System for the CERN PS Accelerator Complex", IEEE Transactions on Nuclear Science, Vol. NS-30, No.4, August 1983.

[3] M.Tyrrell, "The LEP Alarm System", ICALEPCS'91, Tsukuba, Japan, Nov-1991.

[4] R. Billen *et al.*, "Accelerator Data Foundation: How It All Fits Together", ICALEPCS'09, Kobe, Japan, Oct-2009, TUB001.

[5] Z.Zaharieva *et al.*, "Database foundation for the Configuration Management of the CERN Accelerator Controls Systems", ICALEPCS'11, Grenoble, France, Oct-2011, MOMAU004.

[6] Z. Zaharieva, R. Billen, "Rapid Development of Database Interfaces with Oracle APEX, used for the Controls Systems at CERN", ICALEPCS'09, Kobe, Japan, Oct-2009, THP108.

HYPERARCHIVER: AN EPICS ARCHIVER PROTOTYPE BASED ON HYPERTABLE

M. Giacchini, L. Giovannini, M. Montis, G. Bassato, J.A. Vasquez, G. Prete, A. Andrighetto,
INFN/LNL, Legnaro (PD), Italy
R. Petkus, BNL, Upton Long Island, New York, USA. R. Lange, HZB, Berlin, Germany
K. Kasemir, ORNL, OakRidge, Tennesse, USA.
M. Del Campo, ESS-Bilbao, Zamundio, Spain
J. Jugo, University of the Basque Country, Leiola, Spain.

Abstract

This work started in the context of NSLS2 project at Brookhaven National Laboratory. The NSLS2 control system foresees a very high number of PV variables and has strict requirements in terms of archiving/retrieving rate: our goal was to store 10K PV/sec and retrieve 4K PV/sec for a group of 4 signals. The HyperArchiver [1] is an EPICS [2] Archiver implementation engined by Hypertable, an open source database whose internal architecture is derived from Google's Big Table. We discuss the performance of HyperArchiver and present the results of some comparative tests.

EPICS ARCHIVER BRIEF HISTORY

The Channel Archiver [3] is an archiving tool-set for EPICS based control systems. It can archive any kind of record available through Channel Access [4], the EPICS network protocol. Bob Dalesio designed the original index file, data file layout, and implemented the first prototype of Channel Archiver. From then on, many people in the collaboration have been involved on this EPICS extension. The largely used Archiver version is still based on that design and last release is dated on August 29, 2006. That version is based on its own binary file format to archive the PVs. In July 2009 SNS stopped using such version and designed a new archiver. The new version can be engined with two kinds of RDB: MySQL or Oracle. All code has been written in Java and embedded into the Control System Studio (CSS)[5]. CSS can be used to browse the data and to retrieve them. The SNS archiver is based on Oracle RDB in production and MySQL for test purposes; the data base is accessible via Control System Studio (CSS) data browser. This archiver version realized at SNS embeds a Java Archiver into CSS.

TARGET

NSLS2 project at BNL expects a large amount of data that have to be acquired really fast. The goal was storing 10K PV/sec and retrieving 4K PV/sec for a group of 4 signals with a more reliable and safe archiving architecture. The SNS archiver, based on Oracle, seems not fast enough and expensive because of Oracle DB licence fee. The MySQL version has potential scalability concerns, and is designed for a single machine, then heavy processing loads may require expensive hardware and ad hoc solutions. At that time, summer 2009, two approaches were evaluated:

- Rewriting the Rtree and double-linked list embedded DB structure
- Look for a complete replacement of the embedded DB

After an initial period spent in analysing the first solution, we switched to the second one and looked for a completely new technology. The idea came from the observation that the fastest and largest DB available nowadays is Google, whose search engine is based on the proprietary BigTable filesystem. Google never published the details of Bigtable implementation, nor released a licensed version of its filesystem. However, we found an open source product based on similar concepts and technology (Hypertable [6]) and started working on it. Hypertable is a massive, parallel, high performance database, that isn't a RDB nor a SQL DB. During our first tests, Hypertable was only available as part of an open source software project (under GNU2 Licence). Nowadays Hypertable is delivered either in a free version (used for our project) and in a commercial version (by Hypertable Inc.) with payable support. We are using now the 0.9.6.0 version (the last beta release); the roadmap has fixed the first major release 1.0 in December 31st 2011, which will mark the end of the beta period and will introduce a fully functional BigTable implementation.

SYSTEM DESCRIPTION

The system used is a virtual Linux box with 16 CPUs, 10 GB Ram and 3.6 TB of ISCSI disk space on a HP server model DL380G7. A similar system running on a native OS, instead of a virtualized box, would exhibit better performances: we decided, however, to use a virtual machine to make easy cloning the “box” and sharing it with other developers, saving them the time to set-up a complete system from scratch. The virtual layer technology we used is KVM and the underlying operating system is Linux CentOS 5 (a RedHat Linux Enterprise recompiled from source).

The ISCSI disk has been tested separately to be sure it couldn't be a bottleneck for our tests and it showed a R/W

speed of 45 MB/sec; we can assume, therefore, it didn't affect our evaluation.

The bench-test core is an IOC with the following features:

- 10K Record type Analog Input
- Every record has a scan rate 1 sec

The HyperArchiver is configured to insert 10K samples/sec and to extract, from CSS, 1K samples of 4 PVs.

The HyperArchiver configuration is an hybrid system based on MySQL and Hypertable 0.9.5.0.pre6. MYSQL DB is used to store the configuration setup while Hypertable is used to store the Channel data: channelName Status, TimeStamp, Severity, Simple Mode, SimplePeriod, type, and value.

The data read from Channel Access by Channel Archive Engine v.1.2.5 through JCA are concatenated in a string and stored into Hypertable.

The Hypertable schema has only one column named "pv", the rowkey is based on a PV name with the EPICS Time Stamp as follow:

- Rowkey: 100:aiExample.1315639247116
- ColumName: pv
- Data:100:aiExample#c#LOW_ALARM#c#1315639247116#c#MINOR#c##c#jdbc:mysql://localhost:3306/archive#c#Monitor#c#10.0#c#double#c#3.0

Implementation Details

Starting from the original Archiver developed at SNS and based on Java, we made minor changes on various classes; the most important work has been done on package named org.csstudio.archive.rdb which has been designed as shown in Figure 1. The new package named org.csstudio.archive.htrdb2 (Figure 2) has deep

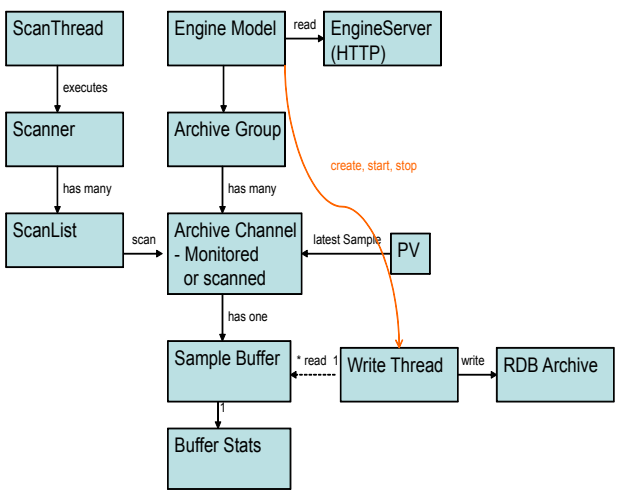


Figure 1: Generic RDB Archive Engine model.

modifications in the class named RDBArchive. Using the Trift client API we added two new classes (Figure 2) :

- HTArchive which contain connections methods;
- HtTableArchiveSerialized which contains methods to manages datas.

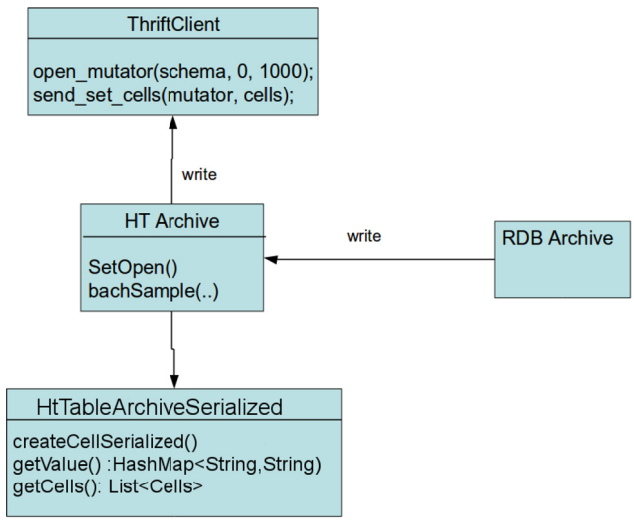


Figure 2: Classe's schema of org.csstudio.archive.htrdb2 package.

The data stored can be readout by dataBrowser2, replacing the package org.csstudio.archivereaderrdb by org.csstudio.archivereader.htrdb2. The most importat modification is on RawSampleIterator class where the Hypertable connection and retrieve task has been realized.

PERFORMANCES

Several test has been carried out at LNL. Using the BNL iocServer we started with a scan period of 0.1 sec for each PV of 10K analog input records. The archive engine seems to be unable to sustain that acquisition rate, so we decide to postpone a more detailed analysis on this and moved a step back using a scan period of 1 sec. for all PVs. With this set-up the average insertion rate is 13 MB/s and the retrieve rate of 13MB/s from CSS. Data were retrieved through the CSS GUI.

CONCLUSIONS

BNL has proposed a common test bench to evaluate the various archiver developments[7]. We used that guidelines for our tests to share and compare our benchmarks with the EPICS community. As mentioned before, the HyperArchiver is still an hybrid system in his internal architecture; it shows anyway good performances and seems a promising research line. Keeping in mind the fundamental spirit of collaboration in Epics community, future steps on this project will be done after a discussion with the laboratories interested on his development. This will help us to better focus on common targets and methods, and obtain the maximum results with the largest benefits for the community.

ACKNOWLEDGEMENTS

Sincere acknowledgements to Bob Dalesio who has made possible the beginning of this research during the stage of M. Giacchini at BNL in 2009.
Many thanks to the great BNL controls team and particularly to R. Petkus and R. Lange for their valuable help. The highly professional collaboration of K.Kasemir from SNS, his great experience and skills were essential. Last but not least, acknowledgements to ESS Bilbao, in particular to M.Campo and Prof. J.Jugo who first trusted on this project and brought it to the production stage at ESS-Bilbao's facilities at Zamundio (Spain) [8].

REFERENCES

[1] HyperArchiver: http://www.lnl.infn.it/~epics/joomla/archiver.html
[2] Epics: http://www.aps.anl.gov/epics/
[3] ChannelArchiver: http://sourceforge.net/apps/trac/epicschanarch/wiki
[4] J.O. Hill: Channel Access: A Software Bus for the LAACS, ICALEPCS 1989, Vancuuver.
[5] Control System Studio (CSS): http://cs-studio.sourceforge.net/
[7] ICALEPCS 11, N. Malitsky, D.Dohan “A Prototype of the Next EPICS Archiver Based on the SciDB Approach”
[8] IPAC 11, M. del Campo, J. Jugo, M. Giacchini, L. Giovannini “EPICS HYPERARCHIVER: INITIAL TESTS AT ESSBILBAO”

IMAGE ACQUISITION AND ANALYSIS FOR BEAM DIAGNOSTICS APPLICATIONS OF THE TAIWAN PHOTON SOURCE

C. Y. Liao#, J. Chen, Y.-S. Cheng, K.T. Hsu, K.H. Hu, C.H. Kuo, C.Y. Wu
NSRRC, Hsinchu 30076, Taiwan

Abstract

Design and implementation of image acquisition and analysis is in proceeding for the Taiwan Photon Source (TPS) diagnostic applications. The optical system contains screen, lens, and lighting system. A CCD camera with Gigabit Ethernet interface (GigE Vision) will be a standard image acquisition device. Image acquisition will be done on EPICS IOC via PV channel and analysis the properties by using Matlab tool to evaluate the beam profile (sigma), beam size position and tilt angle et al. The EPICS IOC integrated with Matlab as a data processing system is not only could be used in image analysis but also in many types of equipment data processing applications. Progress of the project will be summarized in this report.

INTRODUCTION

TPS (Taiwan Photo Source), a third generation 3 GeV synchrotron light facility, featuring ultra-high photon brightness with extremely low emittance which is being in construction at National Synchrotron Radiation Research Center (NSRRC) campus. It consists of a 150 MeV S-band linac, linac to booster transfer line (LTB), 0.15–3 GeV booster synchrotron, booster to storage ring transfer line (BTS), and 3 GeV storage ring.

For optimize machine operation and diagnostics applications, the two-dimensional (2D) beam profile images was used and recorded by cameras, which is widely used in synchrotron light source facility. The beam profile image has extensive information on beam parameters, including beam center, sigma, tilt angle and etc. The fluorescent screens convert the beam flux density as a function of position into a measurable signal, and charge coupled device (CCD) camera for image acquisition, were used in this application. Due to the most of machine parameters in future TPS [1] facility will be accessible as EPICS (Experimental Physics and Industrial Control System) process variables (PVs) [2]. Thus, an analysis tool use the PVs as inputs with ability to calculate and display results in complex ways is needed. In this moment, Matlab was chosen as the candidate computational tools, due to it offers a familiar environment and easy communicated with EPICS PVs. This report presents an EPICS integrated with Matlab as a data processing system which is not only could be used in image analysis but also in other equipments data processing. The hardware configuration and software structure will be summarized in this report.

#liao.cy@nsrrc.org.tw

OVERVIEW OF INFRASTRUCTURE

This infrastructure is developed by using an EPICS IOC integrated with Matlab program to build up a data processing system. For the beam diagnostic application, this system is responsible for the beam profile acquisition from YAG:Ce screens, and used to analysis to find the beam characteristic data. The infrastructure employed can be divided into hardware and software components, as shown in Fig. 1.

Hardware Components

The hardware components include a computer, Gigabit switch, and cameras. The computer system based on EPICS 3.14.10 (Linux, 2.6.18) can act as a soft IOC (Input / Output Controller) which is used to communicate to cameras and collect images data via Ethernet. The Prosilica GC650 VGA-resolution CCD camera with GigE Vision is used which consists of 659×493 (Cell size 7.4 μm) light-sensitive pixels. The cameras have external trigger input port which can be used to receive trigger signal from timing system (EVG/EVR) for periodic data acquisition.

Software Components

The software components are constructed by EDM (Extensible Display Manager, 1.12.xx) and Matlab for cameras controlling and data analysing, respectively. The camera driver EPICS support is built by the areaDetector module [3] which provides a general-purpose interface for area (2-D) detectors at lower-level based on asynDriver [4]. The EDM was used to construct a client OPI for remote control the camera parameters. In this application there are two compiled Matlab programs were used; one is analysis program, the other is display GUI. The Matlab analysis program is continuously running inside the EPICS IOC's PC, through the Channel Access (CA, LabCA) [5] it can read the raw image data from the EPICS database, analyzed it, and then store the results back to the database records. The display Matlab GUI acts as client OPI which can show the analysis results in multiple clients simultaneously. All data are transmitted over the Ethernet network. In this study, Matlab provides a familiar environment and shows easy and efficient ways to reduce the development time and effort. The compiled Matlab programs help us to develop standalone Matlab application without running it in Matlab prompt which can save budget requirement of expensive licence fee. A Matlab runtime environment which is a free download needs to be installed, in order to execute the compiled Matlab program.

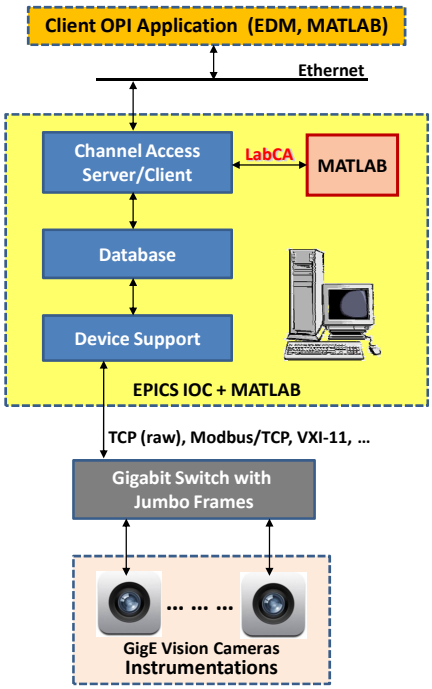


Figure 1: Overview of infrastructure.

FEATURES DEVELOPMENT

In this case, one EDM panel and two Matlab programs were created with difference functionality. The EDM panel which runs in client is used for controlling the camera parameters. The compiled Matlab programs, one is analysis program which runs as a background task in the of IOC's PC for image processing to online calculate the beam size, position and tilt angle; the other display GUI which runs in client for displaying the results and saving data. The detail program designs are shown as follows.

EDM Camera Control Panel (Client Side)

Based on the areaDetector module, using the EDM tool can easy construct a camera control panel as shown in Fig. 2. The camera parameters of exposure time and gain can be configured in this panel. The trigger mode selection include that the Free-Run is for simply monitor the image and Sync-In1 is for synchronization of linac injection (3 Hz). The *Show Image* button could pop-up a window to display the camera image. This EDM panel only offer control and simple monitoring features but do not perform any calculations.

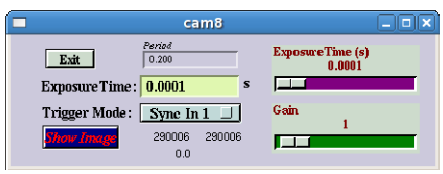


Figure 2: EDM camera control panel.

Matlab Analysis Program (EPICS IOC)

The Matlab analysis program runs in IOC's PC as a background task which can do a complex analytical work to analyze the beam parameters, including the beam center, sigma, and tilt angle. It can specify the region-of-interest (ROI) for clipping the image of each individual camera, and do an optional background subtraction and software multiple exposures. All the analyzed data will store into the EPICS IOC as PVs. The detail flow chart of the program is shown in Fig. 3. In the beginning, program will check which camera is active, then check the corresponding screen is ready or not. If not ready, the program will display some information, such as "No Camera Active" or "Screen is Not Ready". If pass the checking, the program will get raw image data (one-dimensional array) from the EPICS IOC, then go to the image process, including the reshape image array, RIO selection and projection process. Then, processing the image to find the beam center and tilt angle, and fit with a Gaussian distribution for sigma analysis. Finally, writing these results back into specific EPICS IOC PVs, then return to the START. The cycle time can be configured by timer in the program.

The least-squares curve fit, so called lsqcurvefit method was used to process the beam profile, which is one of the optimization toolbox in Matlab [6] which can solve nonlinear curve-fitting problems in the least-squares sense that minimized the sum of squared differences between the measured and predicted data, as shown by Equation (1). Lsqcurvefit is an iterative method which returns results that minimizes the residuals when the tolerances (TolX, default 1e-6) supplied are satisfied. That is, given input data *xdata*, and the observed output *ydata*, find coefficients *x* that "best-fit" the equation *F(x,xdata)*:

$$\min_x \frac{1}{2}\sum_i \left(F\left(x, xdata_i\right) - ydata_i\right)^2 \qquad (1)$$

where *xdata* and *ydata* are vectors and *F(x,xdata)* is a vector valued function. This method can be used to deal with high signal-to-noise levels image.

There are others fitting algorithms can be used, such as the polyfit method, and the moment method. The polyfit method is a non-iterative method which can be used to solve linear problem with faster and accurate analysis, but in this case the data transformed (natural log) into a linear form is necessary. However, it can't deal with the Gaussian distributions with an offset condition, due to the mixed system with non-linear and linear components. The moment method is a computationally simple method to directly calculation the image parameters. For some applications such as image tracking, this method is common to use to analyze large data sets and computationally quicker but sacrifice precision. The three methods comparison for evaluation the computing time, fitting errors and noise tolerance had been described in previous study [7].

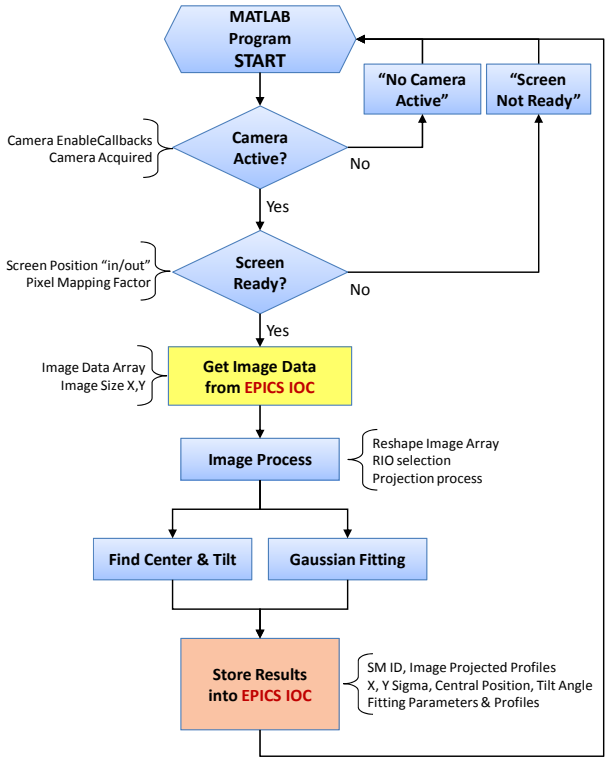


Figure 3: Flow chart of Matlab analysis program.

Matlab Display GUI (Client Side)

The layout of the Matlab display GUI is shown in Fig. 4. The GUI can run in multiple clients simultaneously and read the analysis results from EPICS IOC and display them in the window. The GUI contains six parts: menu, toolbar, control panel, fitting results, projected profile, and raw image. The menu and toolbar provide save data, colormap change, ROI specify, simulation, reset and close program, and zoom in/out functions. In the control panel it contains active the program, 3D viewing, multi-exposure, and background subtract functions. The fitting results area contains sigma and center in the units of pixel and mm, and beam tilt angle. Two directions, horizontal and vertical, of beam projected profiles of raw data and fitting curve are predrilled in two axes. The camera raw image with the colorbar is integrated into the display GUI. The function of export the raw image data and analysis results can be done. It also can create a simulated beam image for the purpose of evaluating the fitting correctness.

The detail flow chart of the program is shown in Fig. 5. In the beginning of the program, users can *Enable* or *Disable* the simulation option. If enable the simulation, the program will generate a simulated beam image then go to analysis process as previous. If disable the simulation, the program will get the analyzed data from EPICS IOC and display the results in the GUI, then return to the START. The cycle time can be configured by timer in the program.

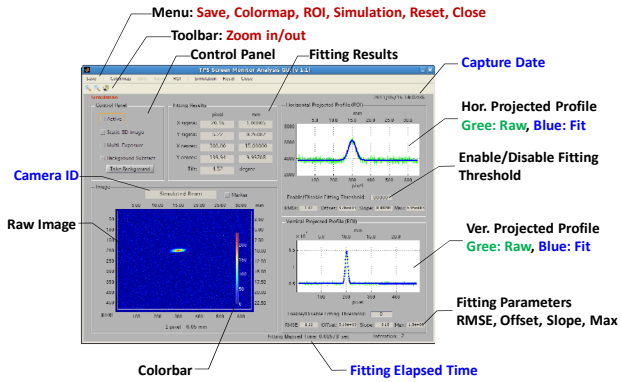


Figure 4: Layout of the Matlab display GUI.

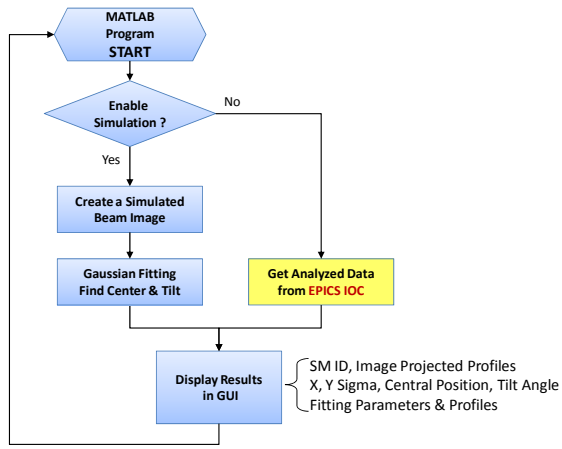


Figure 5: Flow chart of Matlab display GUI.

IMPLEMENTATION IN TPS LINAC

The beam profile of the TPS 150 MeV linac is measured by five YAG:Ce screens [8]. The screen is mounted at 45° against beam direction and driven by a pneumatic driver. The fluorescence light goes out in horizontal direction. The optics consist a refraction mirror bend the light 90° to the 75 mm lens and camera. The resolution of this configuration is ~60 μm/pixel, which is acceptable for TPS current beam size of ~0.5 mm order. The typical layout of the screen monitor is shown in Fig. 6(a). Screen monitor of the test transport line at early commissioning stage is 25 mm diameter YAG:Ce screen with 0.5 mm thickness. This screen is mounted at a manual driven mechanism. This provision solution will be used at early stage of the beam test due to tightly schedule. The typical layout of the screen monitor is shown in Fig. 6(b). The real beam profile, which on the after of gun, was analysis as shown in Fig. 7. The sigma of beam size is in sub-mm order, and tilt angle is also shown in the GUI. The unusual diffraction at position of [X=200 pixel, Y=200~300 pixel] is probably a reflection due to the edge of the YAG:Ce crystal disk. Fortunately, it can be easily ignored by the ROI function. All fitted parameters will be stored as EPICS PVs such that clients can easily access it for further usage.

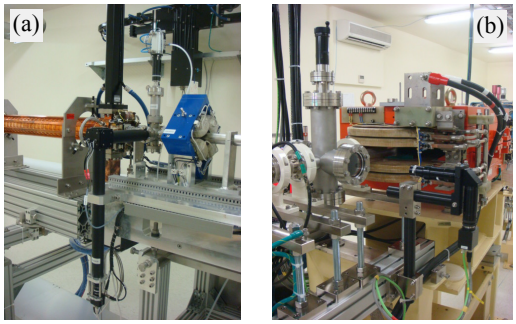


Figure 6: The configuration of screen monitor at (a) linac and (b) test transport line.

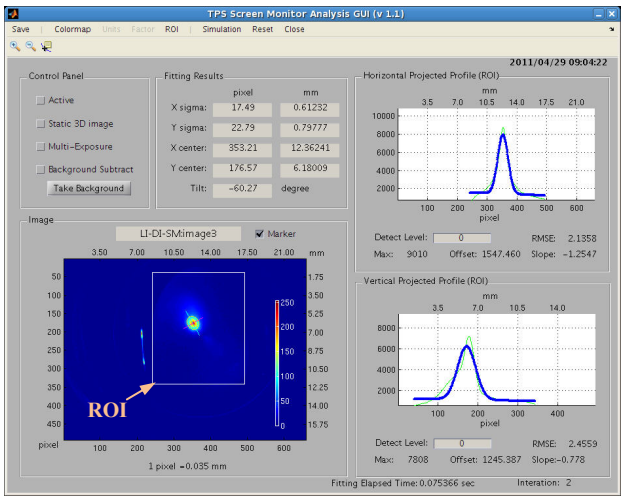


Figure 7: Screen monitor analysis GUI for real beam profile.

TIMING EVALUATION

A timing evaluation of the analysis program was made and summarised in Table 1. The specifications of computer in the experiment for preliminary test are as follows: Intel Xeon 2.26 GHz CPU, 4.0 GB DDR3-1333 RAM. There are two key processes can affect the time required for each image frame. One is read/write data from PVs via EPICS channel access, and the other is fitting projected profiles. The read/write data from PVs, including some conditions and raw image data array, take less than 30 ms. The data process, including two times of fitting process (horizontal and vertical) and tile angle recognition, takes less than 80 ms. Thus, the program requires less than 110 ms per each cycle, which means the maxima processing rate can be up to 9 Hz. This performance is the worst case due to the computer has other tasks to handle at the same time, but this result still acceptable for TPS 3 Hz facility. In future the repetition rate can expect up to 10 Hz rate or higher rate by using a dedicated computer which would be beneficial for many applications.

Table 1: Timing of the Analysis Program

Process	Time Required (ms)
Read/Write data from PVs	< 30
Data process	< 80
Total	< 110

FUTURE DEVELOPMENT

Using EPICS IOC integrated with Matlab as a data processing system is a simple way to develop an application quickly. This solution can be extended to other applications, such as beam lifetime calculation, filling pattern measurement, real-time oscilloscope waveform analysis, synchrotron radiation monitor image analysis, and other equipments data processing. Waveform analysis will be very useful to monitoring the pulse system functionality and stability likes linac microwave system, and pulse magnets system (septa and kickers).

SUMMARY

In this report, we used Matlab to assist EPICS IOC to improve its calculation ability, which will become an alternative solution for complex data analysis. Benefit of this approach can meet expertise of available manpower and more productivity. Compile version of Matlab scripts save budget requirement of expensive licence fee. There are various feature of software have been described here. This infrastructure has been implemented and regularly used on screen monitor image analysis for TPS linac diagnostics application. From timing point of view, there would be no problem running the application on TPS, due to TPS injector repetition rate is 3 Hz. Further studies of other applications of this solution are currently ongoing.

REFERENCES

[1] C.H Kuo et al., Conceptual Design of the TPS Control System, ICALEPCS07, WPPA02.
[2] EPICS (Experimental Physics and Industrial Control System), http://www.aps.anl.gov/epics.
[3] M. Rivers, areaDetector: EPICS software for area detectors, 2011, http://cars9.uchicago.edu/software/epics/areaDetector.html.
[4] M. Rivers, asynDriver: Asynchronous Driver Support, http://www.aps.anl.gov/epics/modules/soft/asyn/
[5] CA (Channel Access), http://www.aps.anl.gov/epics/docs/ca.php.
[6] MathWorks Documentation, lsqcurvefit, http://www.mathworks.com/help/toolbox/optim/ug/lsqcurvefit.html.
[7] C.Y. Liao et al., Beam Profiles Analysis for Beam Diagnostic Applications, IPAC 2011, TUPC146.
[8] C.Y. Liao et al., Diagnostics for the 150 MeV Linac and Test Transport Line of Taiwan Photon Source, DIPAC 2011, TUPD04.

A WEB BASED REALTIME MONITOR ON EPICS DATA

L.F. Li#, C.H.Wang, IHEP, Beijing, China

Abstract

Monitoring systems such as EDM and CSS are extremely important in EPICS system. Most of them are based on client/server(C/S). This paper designs and implements a web based realtime monitoring system on EPICS data. This system is based on browser and server(B/S). Through CAJ interface, It fetches EPICS data including beam energy, beam current, lifetime and luminosity and so on. Then all data is displayed in a real time chart in browser (IE or Firefox/Mozilla). The chart is refreshed every regular interval and can be zoomed and adjusted. Also, it provides data tips showing and full screen mode.

INTRODUCTION

The BEPCII has successfully used EPICS to build up control system. It provides world-widely high energy experiments as well as synchrotron radiation experiments. Because there are many international collaborations, many scientists from outside the IHEP expects to view the status of the BEPCII running. A web based monitoring need to be developed.

In fact, there has been existing monitoring systems on EPICS data including JoiMint, WebCA and CAML.

JoiMint adopts "Java Web Start" to download Java application to local computers and runs it. In order to run Java application, Jre Plug-in or ActiveX is needed to install on client computers.

WebCA is a Web Channel Access client framework based on browser plug-in. It captures EPICS data directly to browsers and displays in text. CAML uses widgets to display EPICS data in graphics. WebCA and CAML can only run in browsers of Firefox/Mozilla.

Since Internet Explore (IE) is popular in China, many people would like to use IE. The EPICS interface to IE is planed, because Internet Explore (IE) uses ActiveX to run web applications while Firefox/Mozilla uses Plug-in. In order to display EPICS data both in IE and Firefox/Mozilla, a web based realtime monitoring system is developed to satisfy the needs of viewing the BEPCII status from all over the word. This system is based on B/S using Flex[1]. The Flex, developed by Adobe, is used to design web user interface, but it needs flash player. Normally, most of computers have been installed with flash player. So, the Flex can run both in IE and Firefox/Mozilla.

CAJ[2] is used as interface instead of JCA, for CAJ is 100% pure Java CA client library and more stable[3]. Through CAJ, Java application in web server can get EPICS data from CA gateways or IOC servers. Blazeds[4] is used as the bridge between Flex and Java while Json is data format transferred from the web server to the browser.

#lilf@ihep.ac.cn

DESIGN AND IMPLEMENTATION

The web based monitoring system is composed of applications in web server and in browser as shown in the figure 1. The parameters of the BEPCII such as beam energy, beam current, lifetime and luminosity runs on PC/Linux as a SoftIOC in control network. The web server runs on PC/Linux in campus network.

The web based monitoring system works as follows:

- The sampling thread connects CA gateways or IOC servers and keeps CA connections in the server's memory.
- Through CAJ interface, the sampling thread gets EPICS data from CA gateways or IOC servers. Then all the EPICS data is saved in the server's memory.
- When visitors view the web page of the monitoring system, the service thread in web server will return the initial EPICS data from the web server's memory to browsers.
- The browser parses EPICS data from the web server and displays it on the web page in patterns of charts and texts.
- By regular interval (10s,30s or 1minute), the browser sends a request to the web server, gets latest EPICS, and then refreshes the charts and texts.

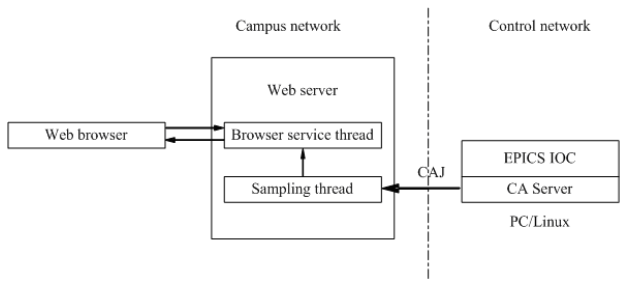


Figure 1: A web based monitoring system of the BEPCII.

Java Applications in Web Server

Tomcat is used as the web server while log4j generates running logs. The Java applications in the web server mainly contain the sampling thread and the browser service thread.

The work flow of the sampling thread in the web server is illustrated in the figure 3. The sampling thread firstly creates CA connections and keeps all of them in the server's memory. Then it gets EPICS data from IOC servers or CA gateway and saves it in the server's memory (as an "application" of the web server). All procedures above executes every regular interval (10s,30s or 1minute). But there are also some differences between the first time and the other times. In the first time, the

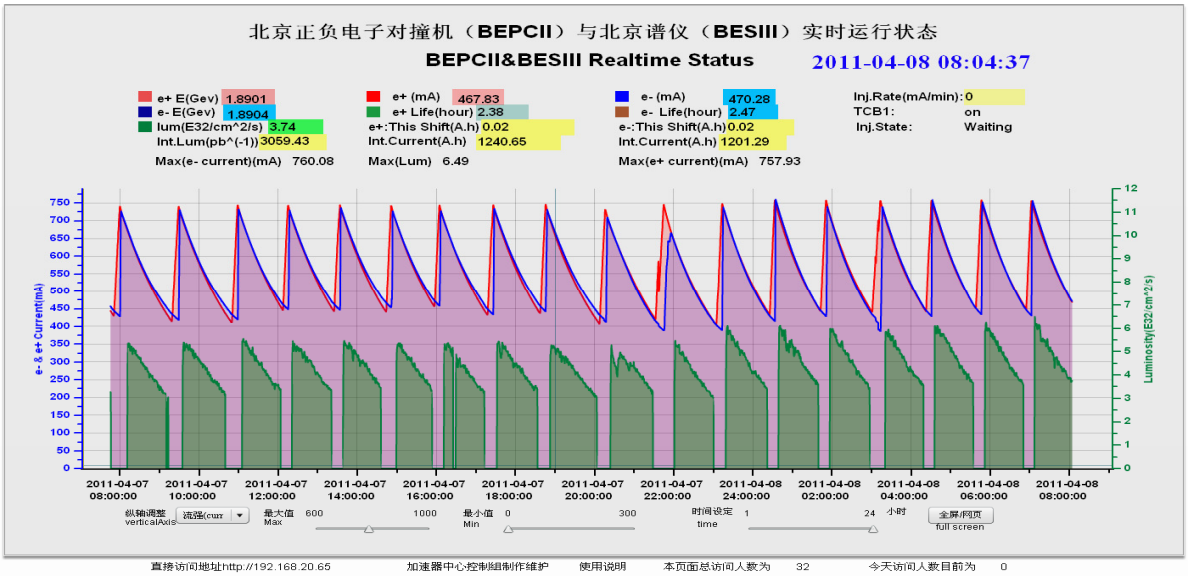


Figure 2: The web page of the status of the BEPCII running.

sampling thread creates CA connections. In the other times, it firstly gets CA connections from server's memory and checks their status. If CA is connected, the thread gets EPICS data directly. Otherwise, it recreates unconnected CA connections and keeps them in server's memory.

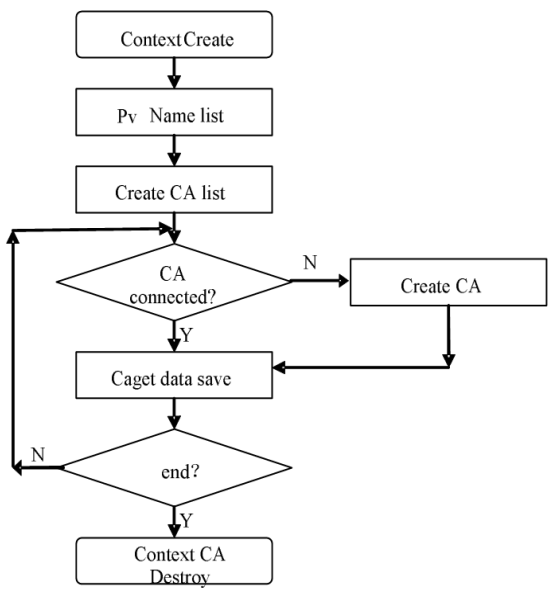


Figure 3: The flow of the sampling thread in web server.

As we know, CA connections take time and CA's UDP broadcasting times by times when recreating CA connections will cause the network traffic. To reduce the time consumed by making CA connections and the network traffic, CA connections should be kept in the server's memory so that the sampling interval will be minimized.

The browser service thread returns EPICS data in format of Json to browser from the server's memory. Then the Flex application in browser displays the data on the web page.

Flex Applications in Browser

The parameters of the BEPCII such as beam energy, beam current, machine mode and so on need to be view by the scientists all over the world. So, we use Flex to implement a web page of the status of the BEPCII running as shown in the figure 2.

This monitoring system adopts Flex and BlazeDS to design the web user interface. The BlazeDS is the bridge between Flex and Java.

When visitors view the web page, the browser sends a request to the web server and gets the initial data of 24 hours. Then, by regular interval (10s,30s or 1minute) , the browser gets the latest EPICS data from the web server and refreshes the chart and texts.

The chart in the figure 2 can be zoomed, adjusted and provides full screen and data tips showing.

Flex applications in the browser mainly include modules as following:

Data Initialization Module

This module gets data in format of Json from the web server and parses the data which is to initialize the chart.

Data Refreshing Module

By method of remote object, data refreshing module receives the latest data from the web server and refreshes the chart and texts.

Timing Module

To fresh the data of the chart and texts, a timer is used to call the data refreshing module by every regular interval (10s,30s or 1minute).

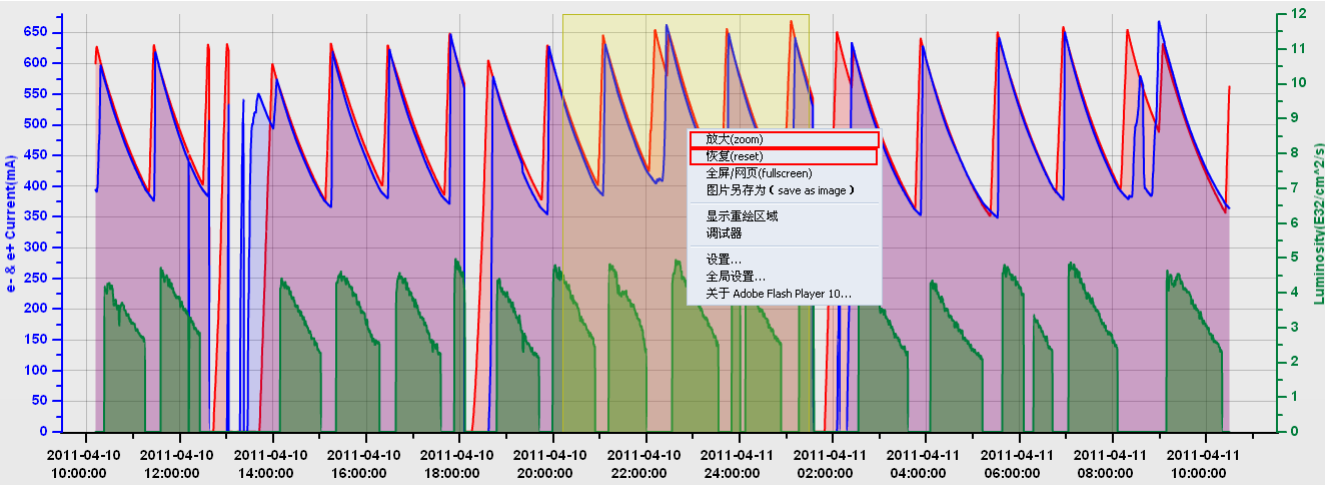


Figure 4: the selected area in the monitoring chart.

Chart Zooming Module

This module is mainly to zoom the chart to look up more detailed information. When a rectangle area in the chart is selected, the corresponding part of the chart is zoomed(see Fig. 4).

Full Screen Module

The web page of the monitoring system can be changed from the web page mode to the full screen mode like movies in media player.

Data Tips Showing Module

When the mouse is placed on the line of the chart, a tip will appears, showing the name, value and time of the EPICS data.

Chart Adjusting Module

By changing the minimum or maximum of axes, the vertical and horizontal axes can be adjusted so that the chart can move up or down, and the total time length of horizontal axes can be decreased.

PERFORMACE TESTS

A web server of tomcat has been built and running almost 2 months without breaking down.

A simple version of the system without special effects (chart zooming ,chart adjusting and full screen) has been available on the web and you can visit the address http://202.122.32.134:8080/RealTimeLineChart/.

Test Environments

- HP Compaq dc7700 (Intel(R) Core(TM)2 CPU 6420 @ 2.13GHz 2G)
- Centos4 linux (synchronizing clock through ntp)
- Tomcat6.0 as web server
- Jdk1.6
- Jprofiler for memory test and thread test
- Apache ab for load test

Performance

The test shows:

- The chart can run in many kinds of browsers(IE or Firefox/Mozilla)
- No EPICS data is lost(once data is lost, it is recorded in the server's log)
- Minimum sampling time=1s
- Concurrent visitors>100
- Web page response time <3s
- Successive working time of server >2 months
- Low load of web server

CONCLUSIONS

This system takes BEPCII running parameters including energy, beam current, lifetime and luminosity as example and fetches these EPICS data through CAJ interface. These parameters can be displayed in IE in a real time curves which can be updated automatically. The web page of the monitoring system is designed by Flex and provides some flex functions including chart-zooming, chart-adjusting, data tips-showing and full screen. On the web page, it's easy for users to adjust the time span and zoom the chart. In order to improve the sampling performance, CA connections stays in the web memory. Since this system is thin based on browser/server, it is very easy to deploy and can be used in other projects.

REFERENCES

[1] http://www adobe com/products/flex html
[2] http //epics-jca sourceforge net/caj/
[3] M Sekoranja " Native Java Implement of channel access for Epics", 10th ICALEPCS, Geneva, Oct 2005, PO2.089-5.
[4] http://opensource.adobe.com/wiki/display/blazeds/BlazeDS

MANAGING INFORMATION FLOW IN ALICE

O. Pinazza, INFN Sezione di Bologna, Bologna, Italy and CERN
A. Augustinus, P. Ch. Chochula, L. S. Jirdén, M. Lechman, P. Rosinský,
CERN, Geneva, Switzerland
G. De Cataldo, INFN Sezione di Bari, Bari, Italy and CERN
A. N. Kurepin, INR RAS – Institute for Nuclear Research of the Russian Academy of Sciences, Moscow, Russia,
A. Moreno, Universidad Politécnica de Madrid, ETSI Industriales, Madrid, Spain.

Abstract

ALICE is one of the experiments at the Large Hadron Collider (LHC) at CERN in Geneva, Switzerland. The ALICE detector control system is an integrated system collecting 18 different detectors' controls and general services. Is implemented using the commercial SCADA package PVSS. Information of general interest, such as beam and condition data, and data related to shared plants or systems, are made available to all the subsystems via the distribution capabilities of PVSS. Great care has been taken to build a modular and hierarchical system, limiting the interdependencies of the various subsystems. Accessing remote resources in a PVSS distributed environment is very simple and can be initiated unilaterally. In order to improve the reliability of distributed data and to avoid unforeseen and unwished dependencies, the ALICE DCS group has enforced the centralization of global data required by the subsystems. A tool has been developed to monitor the level of interdependency and to understand the optimal layout of the distributed connections, allowing for an interactive visualization of the distribution topology.

INTRODUCTION

ALICE is one of the four general purpose experiments installed around the Large Hadron Collider, at CERN in Geneva. It is composed of 18 detectors, each with its own specific technology choice.

The ALICE collaboration includes more than 1000 physicists and engineers from 105 Institutes in 30 countries. Being developed by several groups in parallel, and being based on different devices and protocols, the controls of the different detectors are heterogeneous too.

The ALICE Detector Control System (DCS), integrates the controls of all detectors in a common environment, exploiting the flexibility of the SCADA PVSS software.

INFORMATION FLOW IN THE ALICE CONTROL SYSTEM

The DCS continuously collects large amounts of data from the various detector devices. Thanks to a strong standardization effort at the beginning of the DCS project it has been possible to limit the variety of devices used and this has been very beneficial to the data collection process. For these devices the hardware diversity is managed through the use of the commercial OPC server protocol.

However, for the very important Front-End Electronics part, which has been custom designed for each detector, there is a large diversity of readout interfaces. Many different field busses and technologies are used such as CANbus, JTAG, Profibus, RS232, Ethernet and custom links. This has called for a more elaborate readout architecture based on the definition of a common high level Front-End Device (FED) which overcomes the hardware diversity.

The use of the TPC/IP based CERN DIM protocol for communication provides furthermore the means for abstracting the data flow from the hardware level. Data is mainly transported using the Ethernet but other protocols are also used.

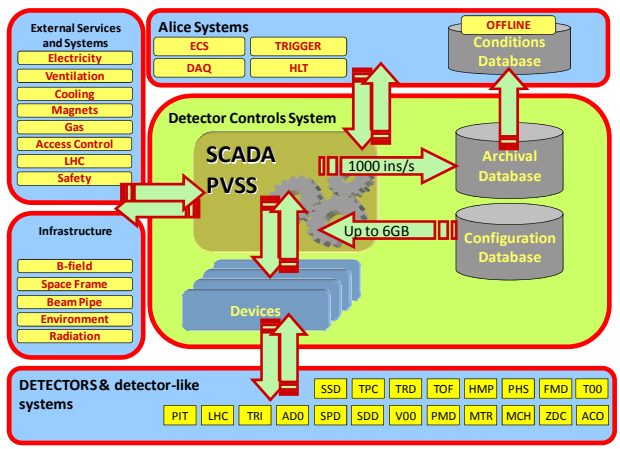


Figure 1: The main data streams managed by the central ALICE DCS.

Amongst the sources of data we count 1200 networked Front-End processors, 300 VME crates and power supplies and 4000 high voltage channels. From the detectors, the ALICE DCS receives data from 180,000 OPC items and 100,000 Front-End devices; eventually, a total of 1,000,000 parameters are supervised by the DCS at a typical rate of 1 Hz (see Fig. 1).

In order to be able to handle such a large amount of data, the ALICE DCS must process and filter the data before they are sent to the archives or transferred to the final users such as the offline analysts and the online displays. More than 100 detector computers represent the

detectors side, while a group of 60 backend servers and an Oracle Database service, which is able to process up to 150,000 inserts/s, compose the central side of the control system (see Fig. 2).

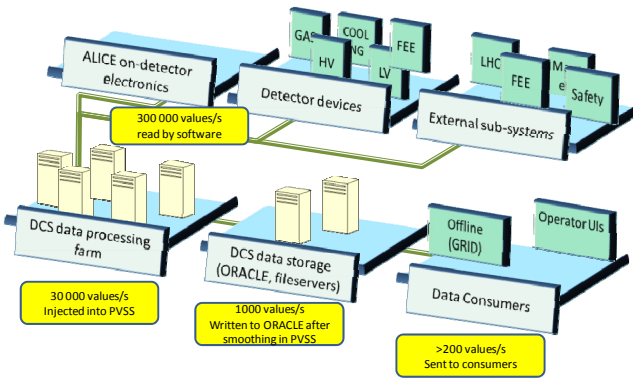


Figure 2. Data flow reduction from the detectors, to the archives and to the final customers.

The data flow is not unidirectional. Up to 6GB of configuration data is uploaded to detectors from the central ORACLE database before a physics run can be started.

Furthermore, the DCS is also the central collector of information of general interest, like the LHC beam data, data from various technical services, environmental data, radiation and magnetic field data, etc.

For example, global environment data is collected and archived and at the same time redistributed to the detectors, to be used for their calibrations or to allow for correlations with their internal variables. Much data is received from the LHC accelerator and managed centrally and distributed to several subsystems in order to define operational states and guarantee safe procedures.

In order to provide a controlled data exchange with the detectors, the ALICE DCS has exploited the distribution properties of PVSS.

The SCADA PVSS is a complex modular system where the different processes (aka managers) can run on a single machine or in a scattered way, i.e. on different machines. In a scattered system, managers run on dedicated machines, in order to distribute the load; in a distributed system, remote, single or scattered, the systems are interconnected through dedicated daemons, so called distribution managers (see Fig. 3); via this connection there is only information sharing and .no sharing of computing resources.

THE ALICE DCS TOPOLOGY

The SCADA PVSS system not only allows to collect, publish and archive the various data. Thanks to its distribution properties and its modular architecture it has been possible to design the ALICE DCS as a distributed system where each detector is a complete control system itself. The detectors have implemented their DCS on more than one machine and are therefore distributed systems themselves.

The smaller detectors have implemented their controls on a few machines which are interconnected in a hierarchical structure. The bigger detectors have implemented their controls over several machines which are interconnected in a mesh, in order to easily share the data inside the detector community. Interactive operation takes place in both cases via a remote user interface on one of the nodes.

A particular group of machines is the LHC cluster, which is responsible for collecting the data from the LHC accelerator and for the publication of ALICE information back to LHC. Some of the these parameters, such as parameters from luminometers and beam monitors, are of direct interest also to the many detectors; they are therefore collected and sent both to LHC and the ALICE detectors. Since its installation, the LHC cluster has represented an interesting testbed for the ALICE distributed topology. A big effort has been devoted to the

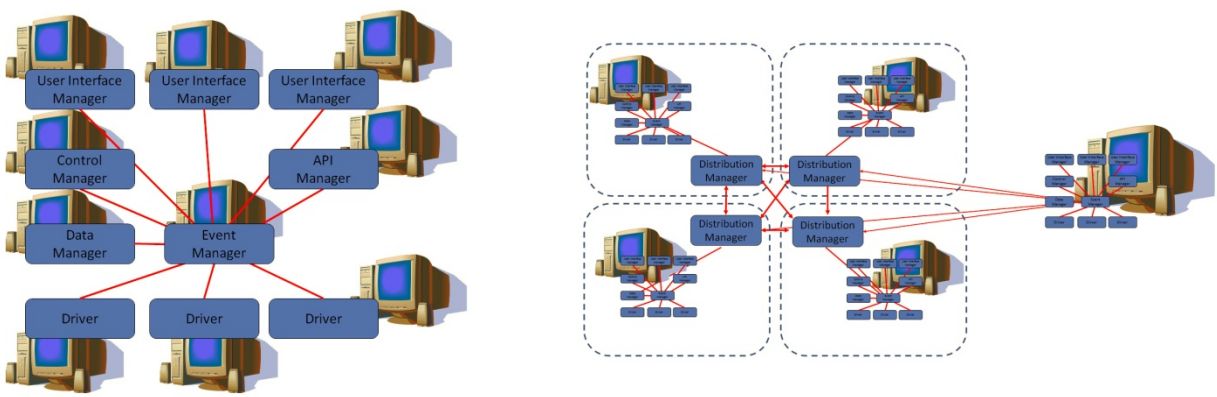


Figure 3: Schema of a scattered system (left) where the PVSS managers of a single project are running on different machines and a distributed system (right) in which the distribution managers handle the connections between remote projects

optimization of the data distribution, trying to limit interconnections, monitoring dependencies, and ensuring the availability and security of the data.

MONITORING THE DISTRIBUTED ENVIRONMENT

Within this distributed scheme, the ALICE DCS has been implemented as a hierarchical system, with the aim of limiting the interdependencies of the various subsystems. All the detectors nodes connect to the central servers (Fig. 4); in this way, ALICE DCS can provide global data, keep the systems synchronized and have a clear monitor of possible interdependencies.

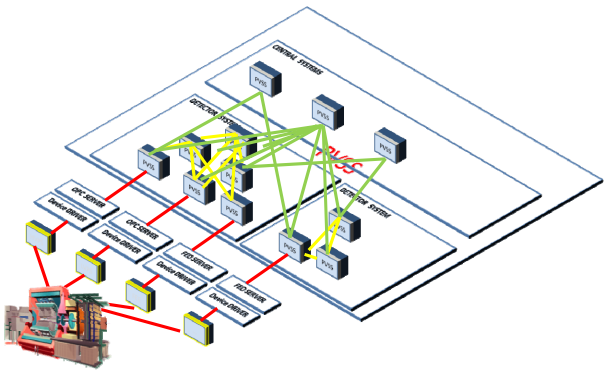


Figure 4. Central servers connect to all detectors nodes. Different detectors are not connected directly.

Connecting remote systems in a distributed environment is very simple. However, this can easily lead to unforeseen and unwished inter dependencies and unnecessary or redundant data flows. Furthermore, a distributed connection can be initiated unilaterally, even without the approval of the target system.

The central ALICE DCS group has therefore initiated a project to map, understand and monitor the data flow between central systems and detectors and between the detectors themselves. The aim is not only to prevent or block unauthorized connections, but mainly to better understand the needs of the different systems and to find the best ways to provide the data they need.

A monitoring tool has been developed which compares a list of authorized, local connections, with the actual connections observed. The authorization list can be maintained and updated from PVSS directly (Fig. 5). The tool has been built using simple PVSS control scripts.

The project provides an interactive graphical representation of the state of connections (Fig. 6). From here, it is easy to assess the degree of interdependency between the systems and to understand the optimal layout of the distributed connections. Used regularly this tool helps to optimize the flow of data between systems and to reduce the risk of faulty interactions between different systems.

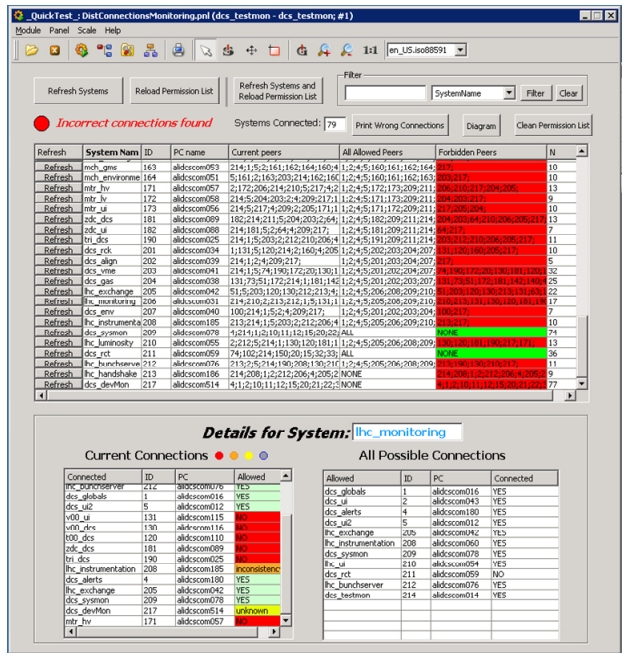


Figure 5. Expert panel to monitor the distributed systems.

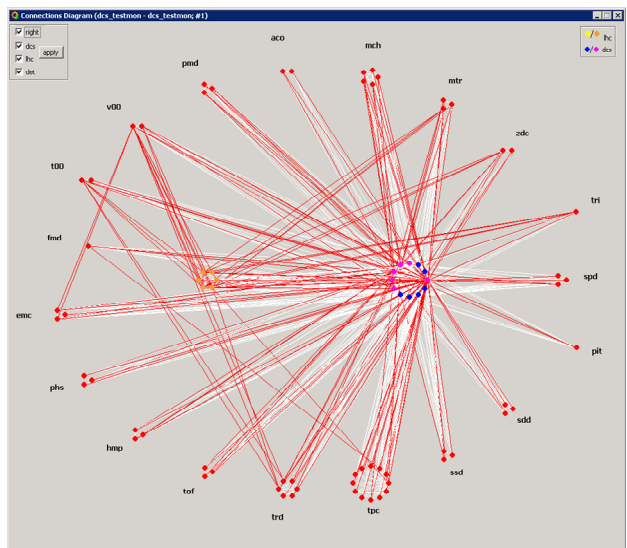


Figure 6. Graphical representation of the ALICE DCS interconnections.

CONCLUSIONS

The ALICE DCS has made a big effort to standardize, simplify and optimize the flow of data between computer systems in the ALICE experiment.

For this purpose a tool has been developed to analyze the network connections between distributed systems. This monitoring tool is raising interest inside the CERN JCOP Framework community, and will hopefully be integrated in a more general monitoring context, to give to the PVSS central developers another tool to optimize and simplify the data flows in the operational networks.

TANGO ARCHIVING SERVICE STATUS

G. Abeillé, S. Pierre-Joseph, J. Guyot, M. Ounsy (Synchrotron Soleil),
S. Rubio (ALBA), G. Strangolino, R. Passuello (ELETTRA)

Abstract

In modern scientific instruments like ALBA, ELLETRA or Synchrotron Soleil, the monitoring and tuning of thousands of parameters is essential to drive high-performing accelerators and beamlines. To keep track of these parameters and to easily manage large volumes of technical data, an archiving service is a key component of a modern control system like Tango [1].

To achieve this, a high-available archiving service is provided as a feature of the Tango control system. This archiving service stores data coming from the Tango control system into MySQL [2] or Oracle [3] databases.

Tree sub-services are provided:

- An historical service with an archiving period up to 10 seconds,
- A short term service providing a few weeks retention with a period up to 100 milliseconds,
- A snapshot service which takes “snapshots” of Tango parameters and can reapply them to the control system on user demand.

This paper presents how to obtain a high-performance and scalable service based on our feedback after several years of operation. The deployment architecture in the different Tango institutes will then be detailed. The paper concludes with a description of the next steps and incoming features which will be available in the next future.

CONCEPTS

The Tango Archiving service is a central diagnosis tool for large scientific instruments. It can be used for daily operation such as:

- Vacuum, temperature follow-up
- Insertion devices follow-up…

But also to analyse complex scientific phenomena:

- Beam orbit stability
- Post-mortem analysis…

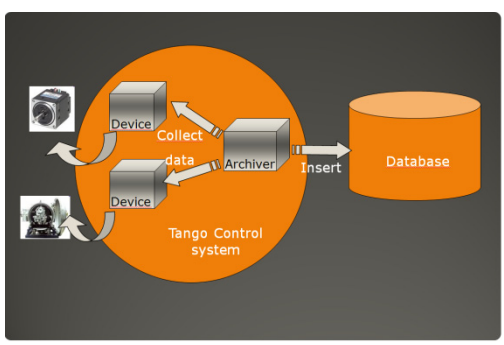


Figure 1: Tango archiving data collection.

The archiving service collects Tango control data (also called Tango attributes) to persist them in databases. The data collection is done by Tango devices called Archivers (see Fig. 1). A complete explanation of Tango concepts is detailed here [1].

All archiving components are based on a Java [4] API (Application Programming Interface), designed with JDBC [5]. Upon this API, a set of modules is rendered:

- Extractor Tango device: service to retrieve inserted data from the database.
- Watcher Tango device: monitor the archiving. Report alarms for not inserted values.
- Manager Tango device: configure, start or stop the archiving.
- GUIs in Swing: End-user interfaces to configure, start or stop the archiving; data plotting of archived values.

The archiving service proposes several data collection policies as detailed here in.

Historical/Temporary DB

The historical and temporary DB services can record data at a fixed frequency. The minimum period is 10 s for HDB and 100 ms for TDB. It is also possible to filter the data to be inserted i.e. to insert data only when it changes or when it goes over some thresholds. Due to the amount of data, TDB tables may be truncated on a regular basis (every 12 weeks at Soleil).

Furthermore, a GUI called Mambo (see Fig. 2) can configure, start or stop the archiving. It is also possible to extract data and plot it.

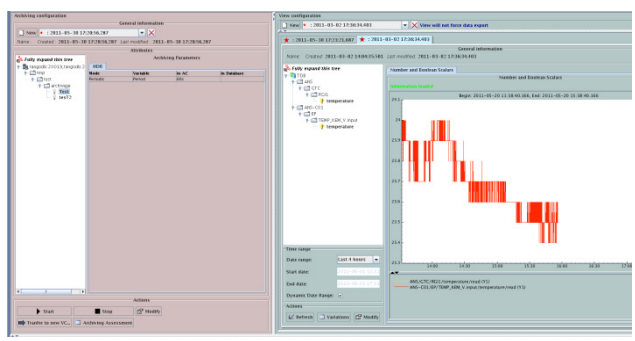


Figure 2: Mambo.

To achieve scalability, a collection of archiver devices may be deployed. Each archiver is in charge of collecting a set of Tango attributes; they may be set up in two different ways:

- Dedicated mode: each archiver collects a pre-defined list of attributes.
- Load-balancing: when a data collection is started, its archiving is allocated to the least loaded archiver.

Snapshot Database

The Snapshot service takes on-user demand "pictures" of a control system for a pre-defined set of Tango parameters (or Tango attributes). The persisted parameters may subsequently be reapplied on the Tango control system.

With the Bensikin (Fig. 3) GUI, the user can define the sets of Tango attributes. For each context, he can launch snapshots, display their contents, and reapply values to the Tango control system.

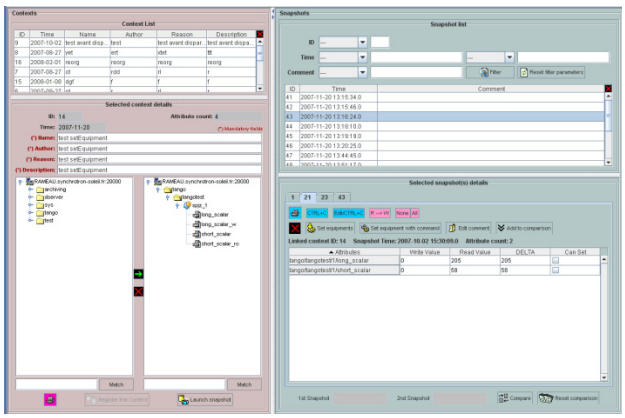


Figure 3: Bensikin.

STATUS

The archiving service is aimed to be used by any institute using Tango as a control system.

The project was initiated by Soleil in 2002. The initial design was jointly done with the ERSF [6] control team who in charge of the Tango core development. All the API, devices and GUIs were developed under the responsibility of Soleil. It was built up during the Soleil construction phase with continuous deployment on production platforms. At this time, the Soleil IT team was constantly under users' pressure to obtain a stable and high-performance system. Due to a lack of resources, most of the development was subcontracted.

Then ELETTRA joined the Tango collaboration at the end of 2003 followed by ALBA in late 2004.

Owing to different contexts and technical backgrounds, the archiving service is not used in the same configuration in all institutes. The actual deployments are described below.

ALBA

The ALBA synchrotron uses MySQL for all databases. Around 8000 HDB attributes are archived at periods between 10s and 1 minute and 2000 TDB attributes are archived at periods between 1s and 1 minute.

95% of the Archivers are deployed on the same host as the Tango archived devices in "dedicated mode".

On beamlines, it is under deployment with a centralized load-balancing mode using virtual servers.

The client applications were developed in Python using the PyTango binding.

Elettra

The TANGO archiving system is used on both Elettra and FERMI@Elettra. Each accelerator is running its own instance of the Archiving System.

Common characteristics to the two archiving infrastructures are:

- MySQL version 5.5.13 with MyISAM indexed and partitioned tables.
- Centralized infrastructure with 30 archivers running for HDB and 10 for TDB
- Management of archiving performed with an Elettra GUI based in QT, MANGO [1],
- Retrieving and data plotting done using E-Giga [1], based on PHP and MySQL.

Elettra figures are:

- 1561 attributes configured in HDB
- 7 attributes configured in TDB

Archiving system for Elettra running on a dedicated hardware based on an Intel(R) Xeon(TM) QuadCore CPU @3.20GHz with 4 GB of RAM and 1 TB of storage capacity on a RAID 6.

FERMI@Elettra figures are:

- 1798 attributes configured in HDB
- 401 attributes configured in TDB

Archiving system for FERMI@Elettra running on a Xen virtual machine using 4 cores of an Intel(R) Xeon(R) CPU E7330 @2.40GHz with 4 GB of RAM and 1 TB of storage capacity on a RAID 6 fibre channel connected external storage.

Both databases are replicated with MySQL replica system on a different machine with the same performances which is used by E-Giga to access data, so the archiving system is not loaded by such kind of requests.

Soleil

The archiving service is deployed on accelerators and beamlines. All databases are Oracle Enterprise Edition 11.1.0.7.0 with RAC (Real Application Cluster) [7] and partitioning options. There are 2 database clusters with each machine having 16 GB of RAM and 2 network interface cards (fiber channel):

- One RAC of 3 machines for the accelerators.
- One RAC of 2 machines for all beamlines.

A Disaster Recovery Plan (DRP) has been established to obtain a high-available service and no data loss:

- A degrade server can take over the RAC.
- Several backup strategies are active, e.g. RMAN (Recovery Manager) [8] does an incremental backup every day and a full back-up every two months. Data restoration is tested twice a year.
- Data are replicated in two buildings, on different media (hard disk and tape).

On top of this infrastructure, the Tango Archiver devices for data collection are configured in "load-balancing" mode and are distributed as followed:
On the accelerators, they are shared out on 4 machines (see Fig. 4 for an overview of the whole infrastructure):

- 270 HDB archivers.
- 300 TDB archivers.
- 1 SNAP archiver.

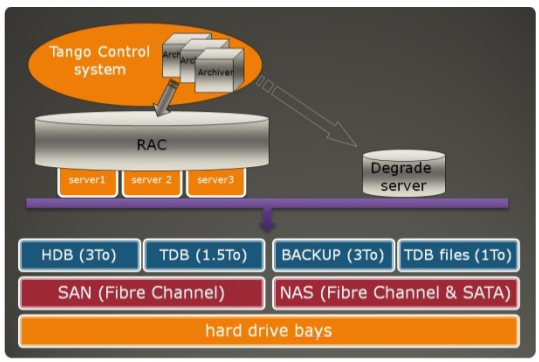


Figure 4: Soleil accelerators archiving infrastructure.

HDB and SNAP is also operational on all 25 beamlines. Some beamlines are archiving up to 350 attributes in HDB; one machine per beamline (with other services installed):

- 5 HDB archivers.
- 1 SNAP archiver.

Table 1 shows some figures for the Soleil accelerators. The numbers are growing constantly as the operators configure the archiving by themselves. Soleil machine operators are planning to collect up to 20,000 attributes in HDB and TDB.

Table 1: Soleil Accelerator Loads

	HDB	TDB	Snap
attributes	10 607	7 527	2700
Insertion/sec	480 (10TPS)	4 200	NA
size	80Go/month	20Go/day	2Go

CHALLENGES

Technical Challenges

Until recently, the main concerns at Soleil have been performance, scalability and stability issues. Various actions have resolved these issues:

- Change the whole infrastructure; install a cluster database with several storage systems.
- The Archiver devices were unstable and their performance was poor. As the code was subcontracted, it suffered from "quick and dirty" fixes, "copy-paste"… To overcome all these issues, the code had to be re-implemented.

The Archivers are now very stable at Soleil. There are no more failures after months of running. However, the code still needs to be cleaned up as has been pointed out by Soleil's quality tool, SONAR [9]. See [10] for how this tool has been used at Soleil.

The current watcher device is only able to report failures, but it is not yet able to diagnose the root cause. Indeed, there may be many reasons for a failed data insertion; for instance the archived Tango attribute may have been renamed, or its device may have been shut down. Consequently, it must be extended to identify and report the origin of a failure.

Data extraction and plotting is pretty basic today. A customisable reporting tool should be integrated in the archiving standard toolkit. The user should be able to simply configure and create reports in any format, such as pdf, csv, and so on. This tool may be also integrated into a scheduler so that machine operators can have an automatic follow-up every shift.

As all data extraction is done manually, all diagnoses are also manual. At Soleil, a project has recently been started to automate the diagnosis of orbit instability with a rule engine [11]. As soon as an orbit deviation is recorded, this tool will automatically search in archived data to identify the causes of instability, such as insertion movements, temperature drifts, power supply misbehaviour, etc.

Collaboration Challenges

A major challenge for Soleil is to fulfil all needs of the Tango community while also remaining responsive to internal user requests, under the constraint of limited human resources. The Tango collaboration has no dedicated resources, so it is at the discretion of each institute to bring some resources to the collaboration. All institutes have different technical backgrounds, it is sometimes difficult to put everything is common as some tools were designed for a specific context, but we must do our best to have the maximum in common.

Effort has recently focused on packaging in order to deliver an easy-to-install archiving system [12]. But there are still many things to improve to efficiently collaborate. The archiving service is now stable, but many evolutions may be added, so a common specification must also be written up. Moreover, this organisation must be also ready to provide support for new incoming Tango users like MAX IV [13].

To conclude, the archiving project is on the one hand a matter of technical challenges, on the other hand it is a major human challenge. It is now a mature project, but it is far from coming to an end as many improvements remain to be done. Besides, this Tango add-on reinforces the added value of the Tango framework.

REFERENCES

[1] http://www.tango-controls.org/
[2] http://www.mysql.com/
[3] http://www.oracle.com/us/products/database/index.html
[4] Java: http://www.oracle.com/technetwork/java/index.html

[5] JDBC: http://fr.wikipedia.org/wiki/Java_database_connectivity
[6] ESRF: http://www.esrf.eu/
[7] http://en.wikipedia.org/wiki/Oracle_RAC
[8] http://en.wikipedia.org/wiki/RMAN
[9] SONAR: http://www.sonarsource.org/
[10] THBHMUST02. Assessing software quality at each step of its lifecycle to enhance reliability of Control Systems. V.Hardion
[11] WEPKS008 Rules-based Analysis with JBoss Drools: Adding Intelligence to Automation. E. De Ley
[12] http://www.tango-controls.org/download/archiving-1
[13] http://www.maxlab.lu.se/maxlab/max4/index.html

FROM DATA STORAGE TOWARDS DECISION MAKING: LHC TECHNICAL DATA INTEGRATION AND ANALYSIS

A. Marsili*, E.B. Holzer, A. Nordt, M. Sapinski, CERN, Geneva, Switzerland

Abstract

The monitoring of the beam conditions, equipment conditions and measurements from the beam instrumentation devices in CERN's Large Hadron Collider (LHC) produce more than 100 Gb/day of data. Such a big quantity of data is unprecedented in accelerator monitoring and new developments are needed to access, process, combine and analyse data from different equipments.

The Beam Loss Monitoring (BLM) system has been one of the most reliable equipments in the LHC during its 2010 run, issuing beam dumps when the detected losses were above the defined abort thresholds. Furthermore, the BLM system was able to detect and study unexpected losses, requiring intensive offline analysis. This article describes the techniques developed to: access the data produced ($\simeq$ 50000 values/s); access relevant system layout information; access, combine and display different machine data.

INTRODUCTION

Beam Losses

More than 3600 *Beam Loss Monitors (BLMs)* were installed around the LHC ring at expected loss locations, in order to measure the *beam losses* in the LHC. Losses are measured every 40 μs, and these basic integration windows are continuously combined into sliding windows called *integration intervals*, ranging from 40 μs to 84 s.

All the data are used in real time: if the losses in one integration interval exceed the predefined *abort thresholds*, the beam is removed from the LHC ring. Some of these data are stored in the LHC *databases (DB)*: this article will only describe the processing of these "offline" data.

The layout of the machine where the loss was produced is just as important as the measured loss itself, and must be understood.

Beam Loss Analysis Toolbox

This article describes the techniques used to access the different databases, to combine the values of losses with the positions of the BLMs and magnets and display them. Each of these functionalities corresponds to a *module*, and they are gathered in the Beam Loss Analysis *toolbox*.

Databases

The *Layout database* holds all the information about the positions of magnets, BLMs and collimators. Its structure is the one of a standard SQL database: several tables hold all related information in different columns.

Every second, for each monitor, one value per integration interval is saved in the *Measurement database* during one week. Some values are stored permanently in the *Logging database* after applying a filter: if the difference between two subsequent values is higher than a delta set per integration interval, the value is saved; if not, one value per minute is saved. The thresholds are logged "on change", and once per day if they don't change.

Both databases have the same structure: a *variable name* `BLM_EXPERT_NAME:LOSS_RSXX` is associated to each integration interval (`"XX"` is its number). It holds two SQL columns: *timestamps* and *values* (see Fig. 1). The structure for the thresholds is the same.

Languages

The database access tool was designed to allow the access, processing and display of data in a fast, user-friendly and easy way. The language used for database access at the time of development was PL/SQL. The Layout database is still accessed with PL/SQL. The main programing and data analysis framework at CERN is ROOT [3], a C++ framework providing classes and a C++ interpretor called CINT.

The choice was made to use Python (2.4) as the developing language (and not CINT), since a true interpreter was desirable. It links easily with ROOT (5.28) objects thanks to the pyROOT project.

DATABASE ACCESS TECHNIQUES

The `database` module accesses data by PL/SQL and Java, and returns them in a format compatible with the other modules, described in Fig. 1. A direct display of the data structure is shown in Fig. 4.

PL/SQL Access

The core of the SQL access class is a ROOT object called `TSQLServer` [3]. Queries such as the time window or variable names are automatically constructed and sent as strings. The data corresponding to the required variables are returned also as strings so the user does not have to change method depending on the data type. SQL access is still used for the Layout database access.

Java Access

The current way to access the measurement databases is through an *Application Programming Interface (API)* written in Java and provided by the data management section.

* amarsili@cern.ch

$$\left\{ \begin{array}{cccc} \text{'BLM\_1'} & \text{'BLM\_2'} & \cdots & \text{'BLM\_m'} \\ \left[\begin{array}{c} \\ (\text{'}t_1\text{'}, v_1) \\ \\ (\text{'}t_2\text{'}, v_2) \\ \end{array} \right] & \left[\begin{array}{c} (\text{'}t_3\text{'}, v_3) \\ \\ (\text{'}t_4\text{'}, v_4) \\ \\ (\text{'}t_5\text{'}, v_5) \end{array} \right] & \cdots & \left[\begin{array}{c} \\ (\text{'}t_6\text{'}, v_6) \\ \\ (\text{'}t_7\text{'}, v_7) \\ \end{array} \right] \end{array} \right\}$$

Figure 1: Original structure of the object for the Logging database, reflecting the DB structure. Dictionaries are represented between curly brackets "{ }", lists between square brackets "[ ]" and tuples between parentheses "( )" The vertical spaces between 2-tuples indicate the absence of data between two times and show that all times are different.

It is able to be called from the command line, thus being more portable: the data are not returned inside one specific application. They are written to the shell standard output.

The API provides the user names and passwords for the Logging and Measurement databases, so the user does not have to. Most of the connection settings are written in a file called `configuration.properties` [4]. Rather than having the user edit this file separately (and for backward compatibility), this file is generated automatically.

The default Python `os` module provides the interaction with the unix shell from Python. The `os.popen` function executes a string as a command line, which allows easy manipulation. It returns a "file iterator", the default object for reading text files. The use of this command line is further described in [4]. The data flow is summarized in Fig. 2.

Data Structure

The object holding the data reflects the structure of the database: dynamic *lists* of immutable *2–tuples* holding the timestamp as a string and the signal as a Python float (see Fig. 1). The mapping is provided by a *dictionary* associating a key (the variable name as a string) to the corresponding value (a list).

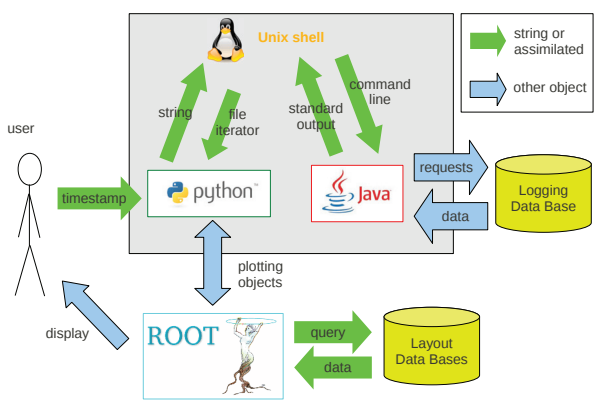


Figure 2: Data flow in the toolbox, from the user to the databases and back.

$$\left\{ \begin{array}{ccc} \text{'name\_of\_graph\_1'}, & \text{'name\_of\_graph\_2'}, & \cdots \\ \left[\begin{array}{c} (dcum_1, v_1) \\ (dcum_2, v_2) \\ \vdots \end{array} \right], & \left[\begin{array}{c} (dcum_3, v_3) \\ (dcum_4, v_4) \\ \vdots \end{array} \right], & \cdots \end{array} \right\}$$

Figure 3: Final structure of "plotdict" object after data combining. $dcum_1$ is the position of BLM_1, and v_1 the signal of BLM_1 at the plotting time. See Fig. 5 for the corresponding graph.

DATA PROCESSING

Sorting BLMs

Currently, one of the main applications of the toolbox is to display a longitudinal loss profile of a section of the LHC at a requested timestamp (see Fig. 5). The names of the BLMs are not displayed, but they contain all relevant information: type of monitor (*Ionisation Chamber (IC)* or *Secondary Emission Monitor (SEM)*), associated beam (B1 or B2), injection or extraction line. The `process` module sorts the BLMs according to these criteria. The names are then mapped with the longitudinal position of the BLM on the LHC, called *DCUM*, which was obtained from the Layout database. The resulting structure is shown in Fig. 3.

Linear Interpolation

One of the features of the Logging database is that no BLM signal is recorded every second (see Fig. 1). A specific second requested by the user may not have been recorded. The choice was made to calculate a linear interpolation between the previous and next entries to estimate the signal at this second. The algorithm in the `process` module progressively checks the requested timestamp against a list of tuples. If the same time is found, the value (2^{nd} element of the tuple) is returned. If not, a linear interpolation is calculated using the next entry in the list.

The algorithm has to do $m \times n$ operations, where m is the number of monitors and n the number of tuples before the requested second. The processing time is dominated by the data access time in the case of the Logging database, when only one second is requested in a time range composed of a few tuples. It is not the case for the measurement database (one tuple per second). If all seconds in the time range are requested, the number of operations becomes $m \times \frac{N \cdot (N+1)}{2}$ where N is the total number of tuples (seconds): the processing time is dominating the access time. In this case, a different algorithm is used with only one index for all BLMs. The number of operations is then only m.

DISPLAY

The whole `display` module relies on ROOT objects, using `TGraph` for the plots and `TCanvas` for the display on screen [3]. Each type of plot has its own class, and they all

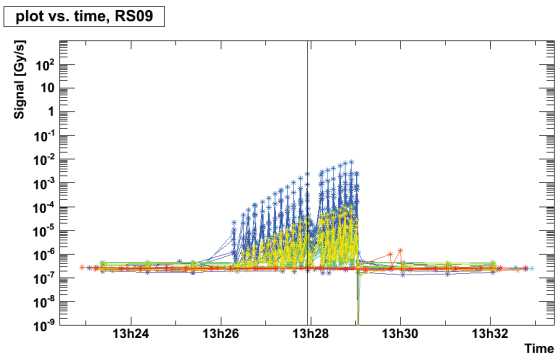


Figure 4: Plot of the data as received from DB, showing the time development of a loss (collimator scraping). Each of the requested variables are given versus time. Note that the values are not synchronised. The vertical black line corresponds to the requested second. The requested plot is shown in Fig. 5.

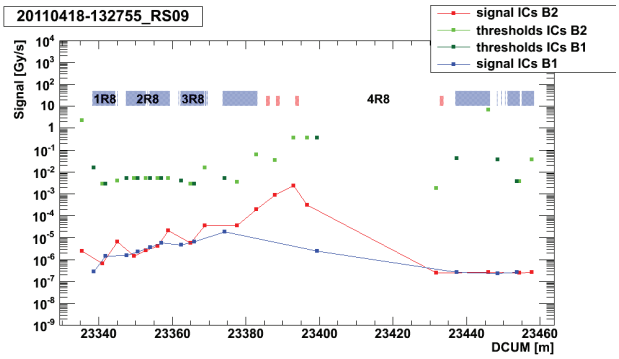


Figure 5: Final display, showing the longitudinal development of a loss (collimator scraping), and the positions of the magnets (blue boxes) and the collimators (red boxes) The vertical axis is the value of loss or threshold of one BLM, and the horizontal axis is its longitudinal position.

share methods to save the plot as seen on screen in a file with an automatically generated and meaningful name.

Time Plot

The time plots reflect the structure of the DB: each graph corresponds to one variable, with points only at recorded times (see Fig. 4). The colours are automatically selected to be as far away as possible from each other. The names are ordered so that similar variables will have similar colours.

Longitudinal Plot

For the longitudinal plot of the LHC, the positions of magnets and collimators are obtained from the Layout database. The value of the threshold is obtained directly from the Logging DB, and is mapped to the longitudinal position similarly to the BLM signals. The final result is shown in Fig. 5.

WRAPPING MODULE

All functions described previously are automatically executed when passing a timestamp and two values of longitudinal position to the loss analysis class in the wrapping module. In its basic behaviour, it:

- finds the corresponding BLMs, collimators and magnets with information from the Layout DB;
- gets the losses and thresholds for the relevant BLMs from the measurement or Logging DB;
- displays the losses versus time (see Fig. 4);
- combines all data to display the loss in the LHC at the requested time (see Fig. 5).

The wrapping module also provides additional methods to:

- save the data set;
- print all the names of the LHC elements in longitudinal order;
- search for a string in the BLM names;
- plot a vertical line at a given longitudinal position;
- plot the ratio signal/threshold for each BLM;
- order BLMs by “closeness to threshold”;
- plot the beam intensity at the time;
- plot the position of the jaws of the displayed collimators;
- plot the signals of the Beam Position Monitors;
- plot the optics in the LHC.

CONCLUSION AND FUTURE IMPROVEMENTS

In this paper, a tool allowing the access, processing and display of data in a fast, user-friendly and efficient way was presented. Data can be requested automatically with a few simple arguments such as timestamp and position, and are immediately available for display or further processing. This tool is now used as a standard database access in the CERN Beam Loss section.

The future improvements include parallel downloading and processing of data, and adaptation to the new vector format for BLM-specific databases.

REFERENCES

[1] All available views, C. Roderick, BE-CO-DM, http://lhc-logging.web.cern.ch/lhc-logging/software/public_data_access_views.jpg

[2] LHC Logging Service, CERN-AB-Note-2006-046, http://lhc-logging.web.cern.ch/lhc-logging/software/default.htm

[3] ROOT class index, http://root.cern.ch/root/html528/ClassIndex.html

[4] Logging Data Extraction Client, Command Line API, https://espace.cern.ch/be-dep/CO/DM/CALS/other/CommandLineManual.pdf

COMPUTING ARCHITECTURE OF THE ALICE DETECTOR CONTROL SYSTEM

A. Augustinus, P. Chochula, L. S. Jirdén, M. Lechman, P. Rosinský,
CERN, Geneva, Switzerland
O. Pinazza, INFN Sezione di Bologna, Bologna, Italy and CERN
G. De Cataldo, INFN Sezione di Bari, Bari, Italy and CERN
A. N. Kurepin, Institute for Nuclear Research of the Russian Academy of Sciences, Moscow, Russia
A. Moreno, Universidad Politécnica de Madrid, ETSI Industriales, Madrid, Spain

Abstract

The ALICE Detector Control System (DCS) is based on a commercial SCADA product, running on a large Windows computer cluster. It communicates with about 1200 network attached devices to assure safe and stable operation of the experiment. In the presentation we focus on the design of the ALICE DCS computer systems. We describe the management of data flow, mechanisms for handling the large data amounts and information exchange with external systems. One of the key operational requirements is an intuitive, error proof and robust user interface allowing for simple operation of the experiment. At the same time the typical operator task, like trending or routine checks of the devices, must be decoupled from the automated operation in order to prevent overload of critical parts of the system. All these requirements must be implemented in an environment with strict security requirements. In the presentation we explain how these demands affected the architecture of the ALICE DCS.

ALICE DCS OVERVIEW

The mission of the Detector Control System (DCS) [1] of ALICE experiment is to provide an overall supervision of the experimental apparatus, ensuring correct and safe operation during the physics data taking and also during the standby periods.

Operating in a continuous 24/7 mode most of the year, the DCS performs hierarchical control of the 20 subdectors of ALICE, supervises common systems and infastructure services and communicates with external systems. Full remote control and monitoring is required for the devices located in the underground areas inaccessible during the beam time.

The ALICE DCS uses a variety of devices supervised by the SCADA software. The devices communicate with the supervisory layer using either ethernet networks, or industrial fieldbuses. Distributed and hierarchically designed supervisory system consists of control applications developed using a commercial system (PVSS) and CERN-developed tools.

User interfaces (UI) at different levels provide experts and operators with convenient control panels, allowing ALICE to be routinely operated by one shifter.

OVERALL DCS DESIGN

The DCS partitioning and control hierarchy follows the logical structure of the experiment. There are 20 subdetectors of different complexities and sizes, ranging from TPC and TRD (10 supervisory control nodes) down to ACO or ZDC with just one control computer.

There are several non-detector projects that provide centralized services (rack control, spaceframe monitoring, access control, global variables, DIM server) or communicate with the external systems (electricity, cooling, ventilation, magnet control, environment monitoring, radiation monitoring detector safety system, LHC services, gas control).

The field layer of the DCS consists of many different types of devices - power supplies (HV and LV supplies of several manufacturers), VME processors, custom made front-end DCS boards, TELL boards used in LHC-related projects, ELMB boards used mainly for the monitoring, PLC controllers and other commercially available or custom made equipment. While a large majority of the devices communicate via Ethernet (processors, boards featuring embedded Linux, but also a majority of the power supplies), there are also several industrial buses in use (CANbus, Profibus, Modbus, RS232, VME/VXI, JTAG).

Most widely adopted middleware communication protocols (running on top of TCP/IP) in ALICE DCS are OPC (industry standard, provides useful smoothing and data reduction) and DIM [2] (CERN-developed protocol - used in the front-end communication, state machine message passing, as well as for the communication with external systems).

The supervisory layer is based on commercial SCADA software (PVSS-II from ETM [3]) and JCOP framework [4] running under Windows (XP and Server 2003). Detector DCS experts developed the control applications for the particular subdector with the support of a small central DCS team that is also responsible for all the central DCS systems and infrastructure.

The front-end DCS boards are controlled by Linux-based applications based on a custom framework developed in ALICE. They communicate with their corresponding PVSS supervisory project by DIM.

The control hierarchy is built by the detector and subsystem developers using the finite state machine mechanism of the FSM package [5] developed at CERN. It allows to create a multilayer hierarchical tree structure distributed over several PVSS projects and control hosts. The detector top nodes are integrated into the overall ALICE control flow supervised by the central ALICE DCS node, which allows to bring the whole detector to the desired state (e.g. READY, STANDBY, OFF) by a single FSM command given at the top. At the various levels, the subsystem's FSM logic agent determines which actions should be executed for a given command and distributes the appropriate commands to the child nodes. It also calculates it's own state from the states of its sub-nodes and reports it to the parent node.

The user interfaces built on top of the control applications (based on PVSS and JCOP framework) allow experts and detector operators to bring the whole detector or its part to a desired state by one mouseclick. One top-level command creates an avalanche of commands propagated down the tree and the flow of corresponding state changes backwards. FSM inclusion/exclusion mechanism combined with the PVSS access control capabilities ensure proper ownership of a given sub-tree (either in the central tree or excluded for the experts).

The alert system based on PVSS datapoint properties and JCOP framework tools provides effective means to detect anomalous conditions so that the operators and experts can efficiently react and bring the detector or a subsystem back to the full health.

DCS COMPUTING

The design of the DCS computing cluster follows the overall DCS partitioning (detectors, central services, external interfaces).

Each detector has at least one PVSS worker node, one operator node running the user interface and most of the detectors have also a dedicated Linux front-end node. Larger detectors distribute the control tasks over several PVSS hosts according to the functionality and/or device types (HV, LV, gas, cooling, front-end) that can be further subdivided into the groups according to the detector layout (side, sector).

The detector top node runs the upper FSM control layer(s), distributes commands, collects states of the various subsystems and communicates with the top ALICE DCS node.

The operator nodes are dedicated for the experts to run their (often CPU and memory hungry) user interfaces independently so that they do not affect the main control applications running on the worker nodes.

Several common projects are running on the dedicated central PVSS nodes, such as rack control, gas control, spaceframe monitoring, global variables, alerts or PVSS access control.

The LHC interface subsystem consists of PVSS hosts supervising tasks like beam conditions and luminosity monitoring or beam injection handshake procedure with the central LHC control centre.

Apart from the described FSM control flow (commands and states) there's much larger flow of the configuration and monitoring data in the ALICE DCS hierarchy (peak rate of 150 000 changes/sec).

The configuration data (originating from the database and custom-formatted files) are sent at the various stages of the setup phase from the control layer to the devices (most of the load goes to the front-end). The monitoring data coming from the devices (via OPC, drivers and PVSS managers) are processed at several levels of the DCS hierarchy using PVSS datapoint objects and can be archived to the database by a specialized PVSS DB manager (about 1000 changes/sec archived). The total size of the data stored in the database is at the level of 20 TB/year (configuration plus archive).

The Linux-based ORACLE database system (housing the configuration database and the archive) is also part of the on-site DCS cluster, as well as several fileservers, bootservers, engineering nodes and other support systems.

Several Windows-based fileservers are used to store and backup the DCS projects, software repository and various tools. Central runtime fileserver hosts certain part of all PVSS projects (such as panels and scripts). Several Linux-based bootservers and fileservers support diskless systems (VME processors, boards based on embedded Linux).

The monitoring of the DCS hierarchy is performed at the level of PVSS (JCOP framework tools), the cluster monitoring is mostly based on Microsoft system monitoring package (SCOM) and Intel toolkit (ISM). In total we have over 200 control system computers, about 100 of them running PVSS applications, 70 serving detector front-ends and non-detector services, the rest belongs to the central services (database, fileservers, gateways etc.).

ALICE DCS NETWORK

Due to the nature of the experimental environment we decided to run the DCS on a private network decoupled from the CERN General Purpose Network (GPN).

Many networked devices used in the DCS lack important security features, others are difficult to update/upgrade. Even if the patches/updates exist they have to be first properly tested, that cannot be fully achieved in the limited lab test setups. The requirement of a continuous and stable operation also disfavors frequent changes at any level.

A CERN-wide policy for the computing and network infrastructure for controls (CNIC) [6] was proposed and implemented that allows to restrict the traffic between the security domains (ALICE, GPN, TN). Only certain vital services running in the CERN computing centre are visible from the ALICE network, as well as those provided by the LHC groups on the Technical network (TN) which is isolated from the GPN as well. The networking infrastructure (routers, switches) and tools (to

create and apply the rules) are provided by CERN IT department. It is also possible to further protect certain sensitive devices (e.g. PLCs) within a security domain, isolating them from all the DCS networked nodes but the few explicitly permitted. The data transfer to and from the DCS network is performed via a dedicated gateway, using the fileservers running inside the two security domains.

Only the central and detector DCS experts and developers are authorized to access the DCS network via an application gateway.

The network is physically divided into several starpoints provided by CERN IT that cover all experimental areas including the counting rooms and experimental cavern. In total we have over 1200 networked devices (600 DCS boards, 250 computers, 200 power supplies).

REFERENCES

[1] P.Chochula, A.Augustinus, L.Jirden, S.Kapusta, P.Rosinsky, "Handling large amounts of data in ALICE", ICALEPCS07, Knoxville, USA 2007.

[2] C. Gaspar,M. Donszelmann, "DIM - A Distributed Information Management System for the DELPHI Experiment at CERN", Proceedings of the 8th Conference on Real-Time Computer applications in Nuclear, Particle and Plasma Physics, Vancouver, Canada, June 1993.

[3] ETM website http://www.etm.at (the product was recently rebranded as "Simatic WinCC Open Architecture")

[4] G D.R.Meyers, "The LHC experiments Joint Control Project, JCOP", ICALEPCS99, Trieste 1999

[5] B.Franek, C.Gaspar, "SMI++ - An Object Oriented Framework for Designing Distributed Control Systems", IEEE Trans. Nucl. Sci., Vol 45, Num 4, p. 1946-1950.

[6] S. Lueders, "Computing and Networking Infrastructure for Controls", ICALEPCS07, Knoxville, USA 2007.

ATLAS DETECTOR CONTROL SYSTEM DATA VIEWER

Charilaos Tsarouchas, S. Schlenker, S. Roe, CERN, Geneva, Switzerland
U. Bitenc, M.L. Fehling-Kaschek, S. Winkelmann,
Albert-Ludwig Universität Freiburg, Freiburg, Germany
S. D'Auria, University of Glasgow, United Kingdom
D. Hoffmann, O. Pisano, CPPM, Marseille, France

Abstract

The ATLAS experiment at CERN is one of the four Large Hadron Collider experiments. DCS Data Viewer (DDV) is a web interface application that provides access to historical data of ATLAS Detector Control System [1] (DCS) parameters written to the database (DB). It has a modular and flexible design and is structured using a client-server architecture. The server can be operated stand alone with a command-line interface to the data while the client offers a user friendly, browser independent interface. The selection of the metadata of DCS parameters is done via a column-tree view or with a powerful search engine. The final visualisation of the data is done using various plugins such as "value over time" charts, data tables, raw ASCII or structured export to ROOT. Excessive access or malicious use of the database is prevented by dedicated protection mechanisms, allowing the exposure of the tool to hundreds of inexperienced users. The metadata selection and data output features can be used separately by XML configuration files. Security constraints have been taken into account in the implementation allowing the access of DDV by collaborators worldwide.

SPECIFICATIONS AND DESCRIPTION

The DDV project aims for implementing a data viewing application for DCS data. The application primarily targets users from the whole ATLAS collaboration allowing to access and display data from database archives. The list of the specifications was the starting point of the development and it went through several revisions driven by development expertise and end-user requests:

- Platform and browser independent project,
- Reasonable application startup time (less than 10sec),
- Small response time to requests (order of second),
- All possible navigation mode options (element_name, alias, description),
- Multiple output formats (chart, table, ascii, ROOT),
- Current configuration in XML format option,
- Database protection mechanisms.

The DDV implementation is based on client-server architecture (see Figure 1). The server part is composed of two main parts, one that deals with data and another one that deals with metadata. The client-server communication is done via the http protocol. The client is composed of two main parts, the request-creation part and the output. The highly modular application offers the possibility of implementation of many interesting features which will be described in the next sections.

DDV SERVER

The DDV server is written in python. It accepts requests, it communicates with the DB, it manipulates data in case it is needed and finally returns back the results.

Metadata Organization The ATLAS DCS metadata is information that refers to an archived parameter (*element_name*) that identifies that specific item in the Oracle tables. Besides, each *element_name* can have a *description* and an *alias* which are more user friendly definitions. The metadata information is copied from the database to an SQLite [2] database cache within the DDV server. The organization of metadata in SQLite file, serves two different purposes. Firstly the SQLite tables are configured to be stored in memory offering in this way the quickest possible response time of metadata queries. Secondly, all queries concerning metadata are performed exclusively inside DDV server keeping the database resources as available as possible.

Data Retrieval In the case of data requests DDV acquires data directly from DB. The server communicates with the DB using the cxOracle [3] extension module that allows access to Oracle databases. The request engine is designed in a way that minimize the DB response time and considers possible changes in the mapping between element_name and alias or comment.

Server Stand Alone Use DDV tool can be used in *batch mode* which offers the option of accessing DB information directly by calling DDV sever (e.g. through a terminal) without the need of a browser or any other graphical environment. The server accepts HTTP requests by using a pythonic framework called cherrypy [4]. As soon as a new URL reach the server, it is decomposed and its parts are used to create a meaningful DB request information.
Requests, GET method:
[server]:[port]/metadata/[queryType]/
[selectedSchema]/[system]/[pattern]
e.g. `http:/pcaticstest07:8089/metadata/element_name/atlas_pvssmdt/ATLMDTPS2/*temp*`
Data requests, POST method:
[serverAddress]:[port]/multidata/getDataSafely
e.g. `http://pcaticstest07:8089/multidata/getDataSafely,queryInfo=atlas_pvssdcs,comment_,CICRackControlLArgY0721A2Humidity,10-10-201012:0,11-10-20100:0`

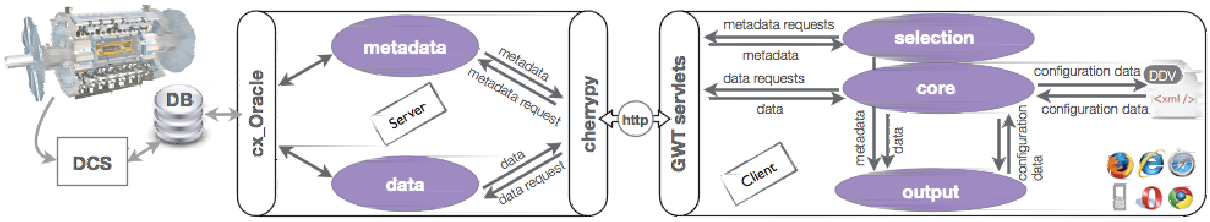


Figure 1: The DDV server-client architecture.

Relational Queries For the cases that users are interested in a subset of data that satisfy some criteria (typical example is the case of a spike) DDV provides a *Relational Queries* mechanism where an accepted range of values can be specified for the selected items. Each item can be configured separately and have its own accepted value region.

Data Base Protection Mechanism Intending to serve a high number of users , DDV aims to be more than an interface to database data. DDV validates and certifies each request with a minimum-response-time cost and finally propagates it to the DB. This protection mechanism is organised in three levels and can schematically be seen in Figure 4. Firstly DDV applies *hard cuts*. Requests with time periods of more than 2 years or including more than 200 items are considered to be potentially dangerous for the DB and are declined. A second protection mechanism performs a light test-request with a time period significantly shorter than the one requested. The number of returned rows is translated to a data rate (Hz) and by extrapolating to the full query time, a decision is made whether this request is acceptable or not. Finally, a third kind of protection mechanism is in place to cover the cases that some unforeseen reason can cause the request to hang. In case DDV does not get any answer from the DB during 20 sec, the request is cancelled before completion.

DDV CLIENT

The client part of DDV is largely based on the Google Web Tool kit (GWT) framework for web applications [5] . The development is done in Java while the result after GWT compilation is a browser independent AJAX-based web application. Keeping the development in a type-based language like Java the testability increases and the debugging of code becomes faster with the help of a Java debugger (e.g. eclipse). The client interface is organized in two main parts. The selection and configuration module (Figure 2) is developed with the use of the GWT framework . With the use of menu items, buttons, tables and other widgets the user enters the request selection criteria and a server request is constructed. The bottom part is independent of the GWT framework and hosts the output plugins.

Selection

Column based browsing DDV provides an easy navigation mode through metadata using column-tree widgets. Since the metadata information is kept in the server in cache, the server response is prompt and in each request the result is returned back to the client in a fraction of a second.

Search Engine The *column lists* is an easy navigation mode. However, the vast number of archived parameters can create difficulties to non-expert users. For such cases DDV provides a *Search Engine* for DCS metadata. The user has to provide one or more strings that will be used as a pattern, separated with the wildcard character '*' . In the DDV server side the search string is translated into a case insensitive SQL pattern which is sent as part of the database request. Apart from the user friendly search engine an *Advanced Search Engine* is in place, that supports regular expressions.

Configuration

Despite the intuitive selection interface of the tool and the optimization of the response time in every action, the selection of the wanted items and the configuration of the final output may take some time. DDV offers the possibility of saving the current configuration to a file. In this way, users can save the current configuration and in a future time upload it again and quickly visualize the wanted information. The configuration file includes general information of the request (e.g. starting/ending time and date), the selected metadata information, the selected output, configuration of the selected output and relational queries information. The configuration file is chosen to be in XML standard which is simple, easy readable and on the same time flexible enough to accommodate future diverse needs.

DDV Outputs

DDV aims to offer to users several output options which satisfy different needs. These outputs are connected with the DDV client with a thin, well defined, interface. There are two main categories of outputs, visual outputs that appear on the browser and output-files to be downloaded to the users disk.

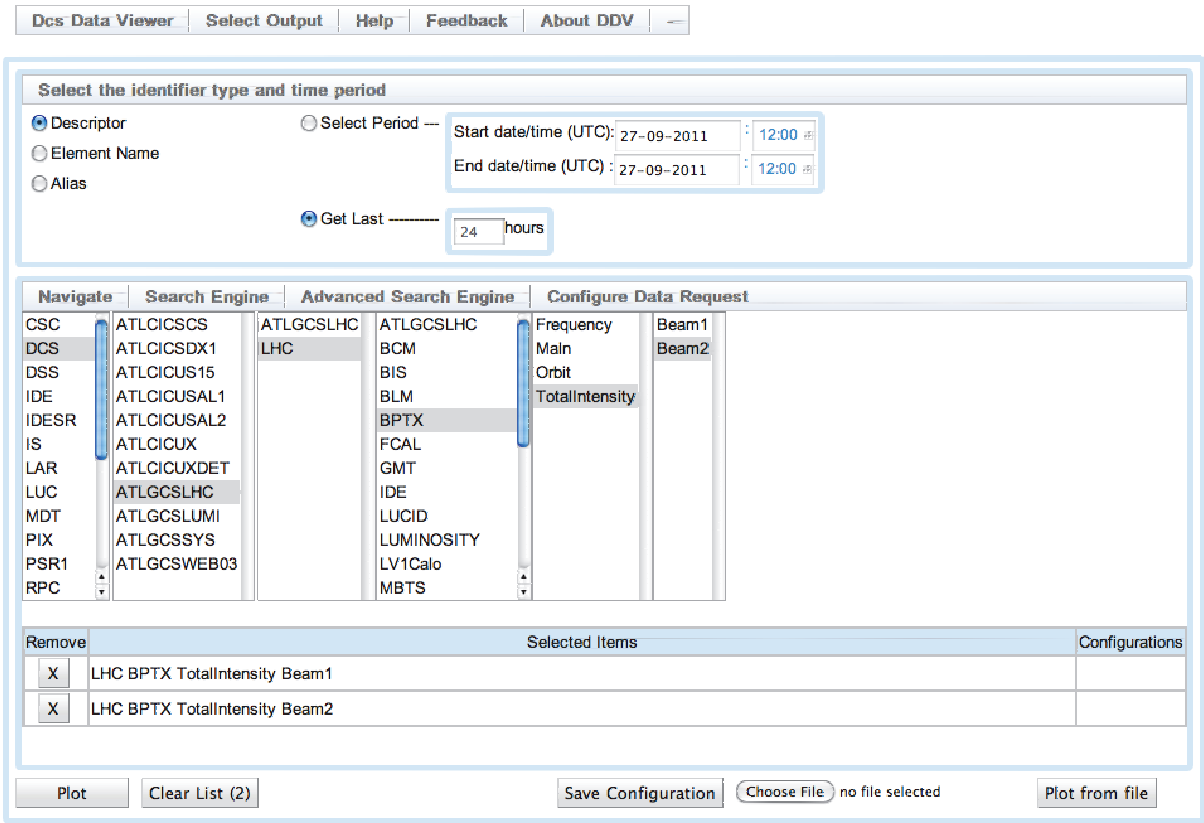


Figure 2: The main browser window interface of DDV. The top part deals with the selection of the navigation mode and the start/end date and time. The middle part offers an easy navigation among the metadata with a column-tree view. The bottom part holds the action buttons like *plot* for a graphical display of data and *save configuration* for saving the current configuration in XML format.

Chart Applet The default output of DDV is a chart (Figure 3), implemented in form of a Java Applet based on the JfreeChart [6] libraries. Depending on the format of the selected data the user can choose to display these as time series, one or two dimensional histograms. The output configuration can be defined either in the main DDV browser interface or interactively in the Applet in a flexible way.

Chart Java Script The Java Script Chart is another output for graphical representation of the data. Concerning the plot options it is more restrictive comparing with the default output but it is more light for the browser. Furthermore, since it is written in Java Script , it can be visualized in environments that do not support java applets (e.g. case of some smart phones).

Table Java Script Data table is a Java Script output that displays the database response in a table. The information held in the table is the archived parameter, the time-stamp and the value. Moreover, the output offers the options of searching and filtering.

ROOT DDV offers the possibility to download the selected data in ROOT [7] format. These files consist of one TTree for every selected data series containing times and values of the contained data points. In this way the user can process the data flexibly in ROOT macros. Besides these trees, there is also a set of predefined TGraphs for the selected data included in the output file.

ASCII Output ASCII file is a simple and highly universal output format. It keeps the information of the archived parameter, the time-stamp and the value.

External Requests

The deliberately modular design of the tool in the initial phase of the development, paid well back in the accomplishment of powerful features. With the use of *External Requests* the user can call DDV by pointing in the same time a configuration file that is pre-saved in the server. All the requested information is kept in the configuration file and in this way a single call accompanied with the configuration file location as a url parameter, is enough to pop up a new browser window with the DDV page and the chart already populated.

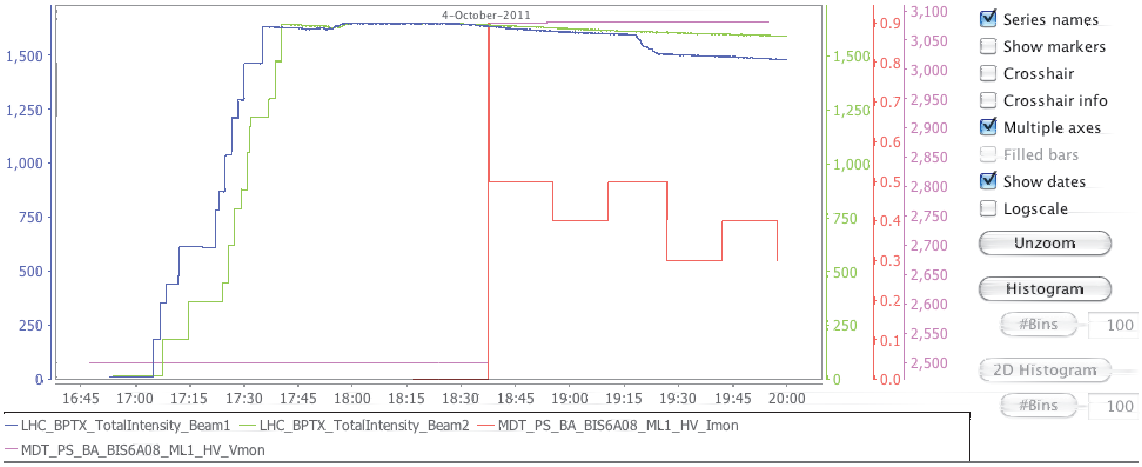


Figure 3: JfreeChart, the default output of DDV. Except of displaying graphically the data, it offers several features like multiple axes, logarithmic scale, markers display, crosshair information, projection of data to 1D and 2D histograms (where applicable).

RESULTS

DDV is officially released as ATLAS central service since the January of 2011. Security constrains have been taken into account in the implementation allowing the access of the application by collaborators worldwide. Currently the number of DDV users is about 150 and it grows constantly.

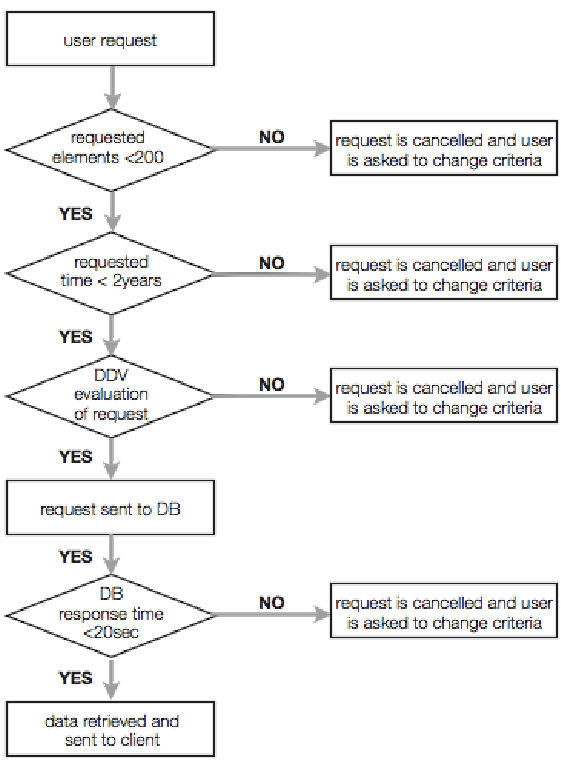


Figure 4: DB protection mechanism.

REFERENCES

[1] A. Barriuso Poy et al., "The detector control system of the atlas experiment", JINST 3 (2008) P05006.

[2] SQLite documentation, http://www.sqlite.org/docs.html.

[3] cxOracle Documentation, http://cx-oracle.sourceforge.net/html/index.html.

[4] CherryPy Documentation, http://docs.cherrypy.org/stable/.

[5] GWT API Reference, http://code.google.com/webtoolkit/doc/latest/RefGWTClassAPI.html.

[6] David Gilbert, JfreeChart documentation, http://ktipsntricks.com/data/ebooks/java/jfreechart-0.9.1-US-v1.pdf.

[7] ROOT Documentation, http://root.cern.ch/root/doc/RootDoc.html.

THE PSI WEB INTERFACE TO THE EPICS CHANNEL ARCHIVER

Gaudenz Jud, Andreas Luedeke, Werner Portmann. PSI, Villigen, Switzerland

Abstract

The EPICS channel archiver is a powerful tool to collect control system data of thousands of EPICS process variables with rates of many Hertz each to an archive for later retrieval.

Within the package of the channel archiver version 2 you get a Java application for graphical data retrieval and a command line tool for data extraction into different file formats. For the Paul Scherrer Institute (PSI) we wanted a possibility to retrieve the archived data from a web interface. It was desired to have flexible retrieval functions and to allow interchanging data references by e-mail. This web interface has been implemented by the PSI controls group and has now been in operation for several years. This paper will highlight the special features of the PSI web interface to the EPICS channel archiver.

INTRODUCTION

The EPICS channel archiver is used at different facilities at PSI, e.g. for the Swiss Light Source (SLS), the Swiss Free Electron Laser (SwissFEL) injector test facility, the medical cyclotron (Proscan) and others (Fig. 1). Several ten thousand process variables are archived at SLS and for the growing SwissFEL project the number is nearly twenty thousand process variables already. The arching of waveforms becomes increasingly important for the SwissFEL project. Currently more than hundred waveforms are collected at 10 Hertz.

To gain most benefit of the archived data, a simple retrieving tool is essential. It has to be taken into account that not only the machine and experiment experts, but also our maintenance staff are using the Archivers and are dependent on an easy retrieval of the archived data.

Taking everything into account, we decided that it should be possible for everybody to retrieve channel archiver data from every computer standing around. The retrieval has to be possible with a minimum of software installations. A web browser is a tool contained in the standard installation of every PC at PSI. Thus we decided to write a web-based server application which is very intuitive and can be learned to be used within minutes by persons, who are no computer experts. The use of predefined data sets was important as well as the possibility to analyze waveforms. Some export formats should allow the users to accomplish further works on the data according to their wishes.

Process variable names are often rather cryptic. Therefore it is possible to define descriptions for EPICS process variables in a dictionary, which are stored in a database. The descriptions can be used fully equivalent with the process variable names: you can either search for process variables or descriptions, and in the data plots, you can decide whether to use the original name or its description in the agenda and for the y-axis labels.

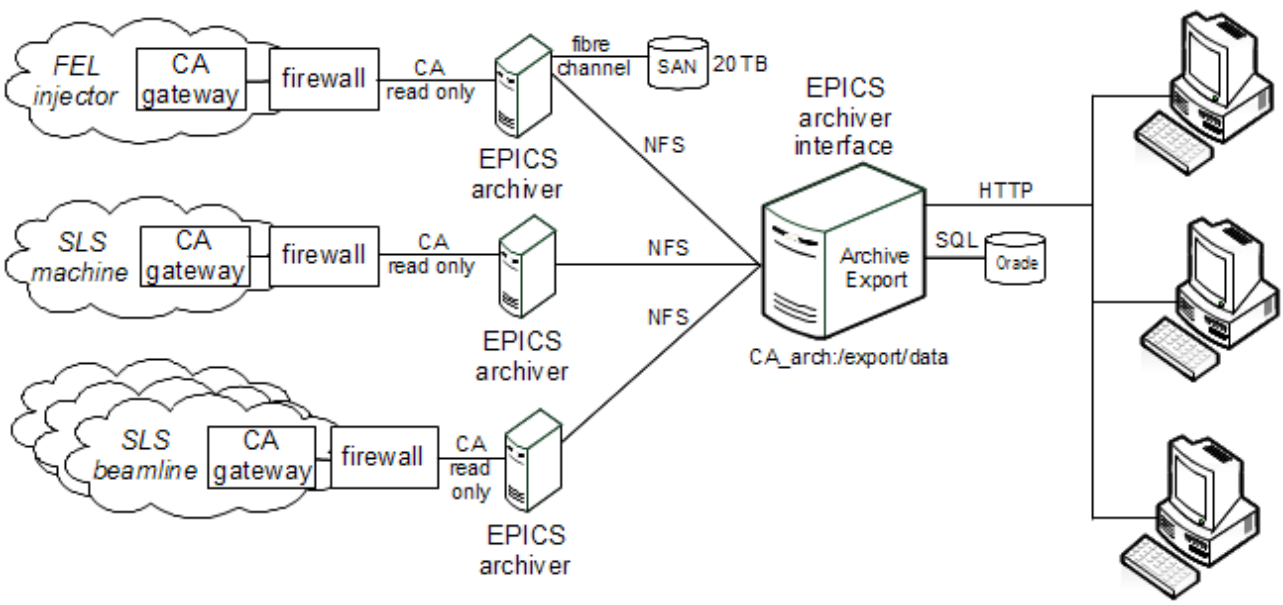


Figure 1: Overview of the EPICS channel archiver realisation at PSI.

GRAPHICAL USER INTERFACE

Scalar Data Plot

Fifty different EPICS archive processes, so-called archive engines, are running currently at PSI. When calling the main page of the archiver data interface, an overview of all these engines is given.

On this main page one can also find a navigation bar on the left. One link of it is **predefined views**, which leads to an overview of several collections of preselected channels, which can be plotted by one click. These predefined views are extremely helpful to examine collections of records routinely (Fig. 2).

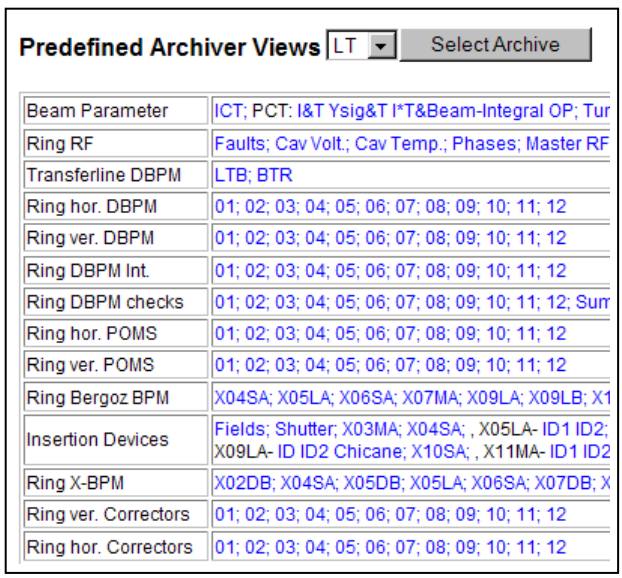


Figure 2: Partial view of predefined views.

If the user decides to select one of the archiver engines displayed at the main page, he/she is lead to a page, were the channels of interested can be entered in a text field.

Alternatively channels can be searched by means of regular expressions. The result of such a search is a list, from which the desired channels can be selected.

The next page provides handles to adjust the time range and the formatting of the axis, buttons to process the data for scalar or waveform display or data export, and several more options to format the output (Fig. 3).

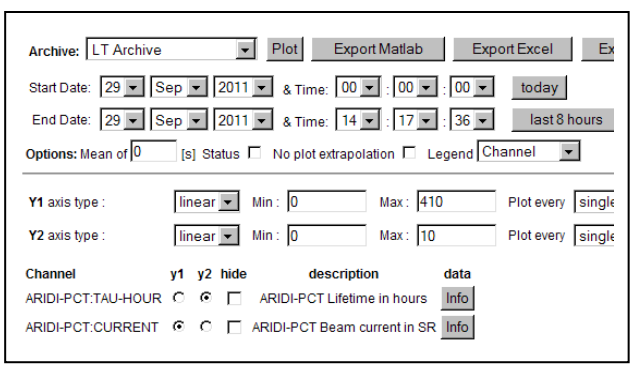


Figure 3: Partial view of the adjustment page.

A plot command with the adjustments in Fig. 2 results in the plot shown in Fig. 3.

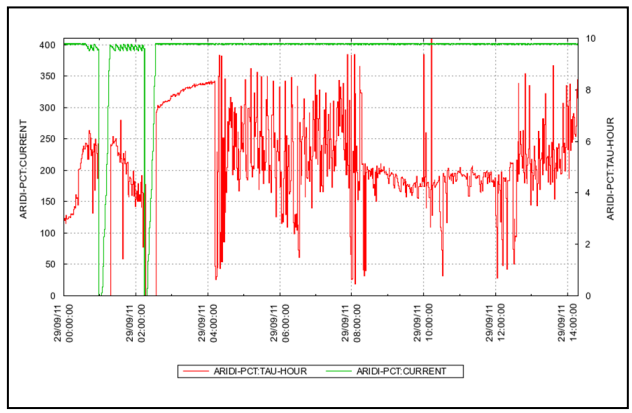


Figure 4: Ring current plot with parameters from figure 3.

Waveform Data Plot

Analogue signals from an AD converter card might be stored as waveforms to achieve a better time resolution. The low level RF group engaged in the development of an injector to be used for the SwissFEL project uses a lot of waveforms. Therefore it is inevitable to provide a tool to analyse the waveform data. A data viewer written in Java was available to perform such analysis.

The java tool was a client application but the users wished a possibility to look at their waveforms data in the same tool as they use for other data in the channel archiver. Therefore we added a waveform viewer on the web server. A lot of ideas were taken from the Java data viewer. The adjustment page of the archiver web interface allows now selecting a waveform display as an option. Then a special page devoted to control the output of the waveforms opens (Fig. 5).

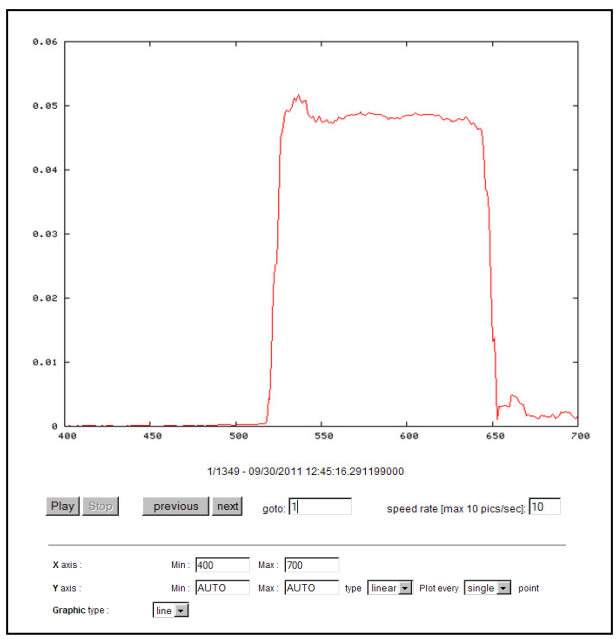


Figure 5: Waveform plot with selectable options.

All waveforms archived during the selected period are created as a sequence of plots. By pressing the play button the sequence is displayed as an oscilloscope display. The speed can be chosen between 1 and 10 waveforms per second. One can step forward and backward plot by plot or choose or choose a dedicated plot within the sequence. The plot can be filled or outlined; the x- as well as the y-axis can be selected. Otherwise an automated wide and height is chosen respectively.

Archiver Data Export

Included in the EPICS channel archiver package version 2 is the command line tool *ArchiveExport*. While useful as a debugging tool, it might be cumbersome to be used for routine data analysis. The data retrieval with our web interface is still done with *ArchiveExport*, but the graphical user interface facilitates its use. All user inputs are translated into a command line using *ArchiveExport*. The retrieved data are processed with the graphical *gnuplot* package to provide the desired plot. The data can be exported either as text, excel sheet or in matlab format.

Uniform Resource Locator (URL)

When using the archiver web interface, one might want to communicate the created plots to somebody else. To fulfil this task, the archiver web interface has an extremely useful behaviour. All features of the interface pages are contained in the URL. Therefore it suffices to copy the URL of a page of interest and send this URL to a recipient. Opening this URL shows exactly the same content as the sender was looking at.

The advantage of interchanging data references as URL compared with interchanging the data itself is many fold. The main benefit is that the recipient can refine the data: he can easily expand the time range or add other process variables in order to correlate events. At the same time the usage of URLs simplifies the access for the user, since he only needs to learn one tool to handle the data retrieval and display. And it saves of course bandwidth, not to interchange the full data of interest by e-mail.

It is possible to go even a step further and to create the URL in another application. This allows web applications to link to the archiver web interface for the display of specific data, with defined formatting for a desired time range. The Operation Event Logging System used at the SLS [1] utilizes this feature of the channel archiver web interface. It creates for each beam event links to related data. For example in case of a beam outage it creates a link to the beam current plot during the outage period, or for each orbit feedback failure it generates a link to the orbit data during the time of the failure.

TECHNICAL REALISATION

Hardware

Using the web server solution to retrieve channel archiver data imposes a powerful hardware. The server hosting the web interface to the channel archiver is a HP Bl460c G6 Proliant Blade Server with 2x quad-core E5506 @ 2.13GHz CPU, 24GB RAM and 1GB/s network. The data are stored on a hardware raid connected with a 4 GB/s fibre channel connection.

Software

The operating system of the server is Scientific Linux SL release 5.1 (Boron) [2]. We choose the development environment Eclipse Galileo [3], used with the Zend Engine v2.1.0 [4] and Xdebug v2.0.5 [5]. The web server is an Apache 2.2.3 [6], PHP used to develop and run the archiver interface has version 5.1.6 [7]. JavaScript is used to perform the control of the waveform output on the client side. The export of data into Matlab format is performed without using the Matlab program. We used the matio-1.3.4 C library [8] to program the Matlab export function. Calling this c module inside PHP enhances the conversion speed remarkably.

SUMMARY

The PSI archiver web interface for the EPICS channel archiver has proven to be a reliable, efficient and effective tool for the access and display of the archived control system data. It is easy to use; it does not depend on other client software than a normal web browser and it provides high flexibility to retrieve the data. Many useful applications derive from its ability to encode all retrieval and display parameters in a URL. This allows other web applications to link to specific data use references to the archiver web interface, instead of incorporating their own measures of data retrieval from the archiver.

ACKNOWLEDGEMENT

The PSI archiver web interface is the successor of the CGIExport tool, which was part of the previous version of the channel archiver [9]. That tool was in use at PSI, too. It was discontinued in version 2 of the channel archiver. We therefore reengineered a new tool with the same layout and functionality. Even the URL interface was kept compatible, but has been extended later for the new functionality of the PSI archiver web interface.

REFERENCES

[1] A. Lüdeke, “Operation event logging system of the Swiss Light Source”, Phys. Rev. ST Accel. Beams, **12** (2009) 024701.
[2] Scientific Linux, http://www.scientificlinux.org
[3] Eclipse galileo, http://www.eclipse.org
[4] PHP und ZendEngine, http://www.zend.com
[5] Xdebug Debugger and Profiler Tool for PHP, http://www.xdebug.org
[6] Apache Software Foundation, http://www.apache.org
[7] PHP: Hypertext Pre-processor, http://www.php.net
[8] MAT File I/O Library, http://matio.sourceforge.net
[9] K.U. Kasemir, L.R. Dalesio, “Overview of the Experimental Physics and Industrial Control System (EPICS) Channel Archiver”, ICALEPCS’01, San Jose, 27-30 Nov. 2001, pp THAP019

ASYNCHRONOUS DATA CHANGE NOTIFICATION BETWEEN DATABASE SERVER AND ACCELERATOR CONTROL SYSTEMS*

Wenge Fu, Seth Nemesure, John Morris, Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

Database data change notification (DCN) is a commonly used feature. Not all database management systems (DBMS) provide an explicit DCN mechanism. Even for those DBMS's which support DCN (such as Oracle and MS SQL server), some server side and/or client side programming may be required to make the DCN system work. This makes the setup of DCN between database server and interested clients tedious and time consuming. In accelerator control systems, there are many well established software client/server architectures (such as CDEV, EPICS, and ADO) that can be used to implement data reflection servers that transfer data asynchronously to any client using the standard SET/GET API. This paper describes a method for using such a data reflection server to set up asynchronous DCN (ADCN) between a DBMS and clients. This method works well for all DBMS systems which provide database trigger functionality.

INTRODUCTION

In normal database server and client operation, the server side stores the data and makes it accessible to clients. Clients dominate the data changes by submitting requests to the server. There are many instances where a client may want notification when the data has changed on the server side by another client. There are two known ways to accomplish this task. One way is to use a database server driven data change notification (DCN) technique while an alternative approach is to have the client synchronously poll for data. The downside to client data polling is an increased (and potentially unnecessary) number of network transactions in addition to an increase in database server side work load. Additionally, depending on the data polling frequency, there may be a delay between the time that the client becomes aware of an update to data and the time the update actually took place. By contrast, database server side DCN provides the client with asynchronous (immediate) updates.

The distributed DCN message structure can be in the form of the actual changed data or can be a form of meta data (e.g. the column in the database table that has changed including whether or not the change was due to a database insert or update). In recent years, many commonly used DBMS (Database Management Systems) such as the Oracle database server[1], Microsoft SQL server[2], IBM DB2 server[3], and Sybase ASE[4] server, provide DCN support using different approaches. There are advantages and disadvantages to each DBMS system's DCN feature. Typically, in order to take advantage of a DBMS provided DCN feature the user may need to:

- Set up a data change notification handler on the server side for target data tables or target data columns, and publish data changes (or event or meta data of the data changes without actually sending the changed data) through the handler;
- Set up a data change notification receive handler on the client side and register (or subscribe) it with the listening DBMS DCN objects that will receive the data change notification (or events and meta data of the data changes) .
- Keep an open (TCP or UDP) network connection between client and database server.

Setup of DCN registration, data change publishing, and receiving DCN may differ between various DBMS to DBMS systems. At a minimum the basic steps listed above must be taken into account. Some database server side SQL programming and client side application programming may also be required to make the system work. This process can be tedious and may require additional effort to support different clients.

In typical accelerator control systems like the RHIC-AGS system running at Brookhaven National Laboratory, there exist common techniques for data communication between hardware and software. Some of these include software interfaces such as ADO[5] (Accelerator Device Object), CDEV [6] services, and EPICS[7] (Experimental Physics and Industrial Control System). These interfaces can be used to setup reflection servers to transfer asynchronous DCN messages from a database server to client without a direct connection between client and database server. This greatly simplifies the setup of the DCN process.

By taking advantage of existing accelerator control system software using the CDEV service, an example of a simple and robust method for setting up a database DCN system is presented. This method works well with any DBMS system which supports database triggers.

SETUP OF DATABASE ADCN

The ADCN system consists of three parts as shown in the Figure 1.

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy.

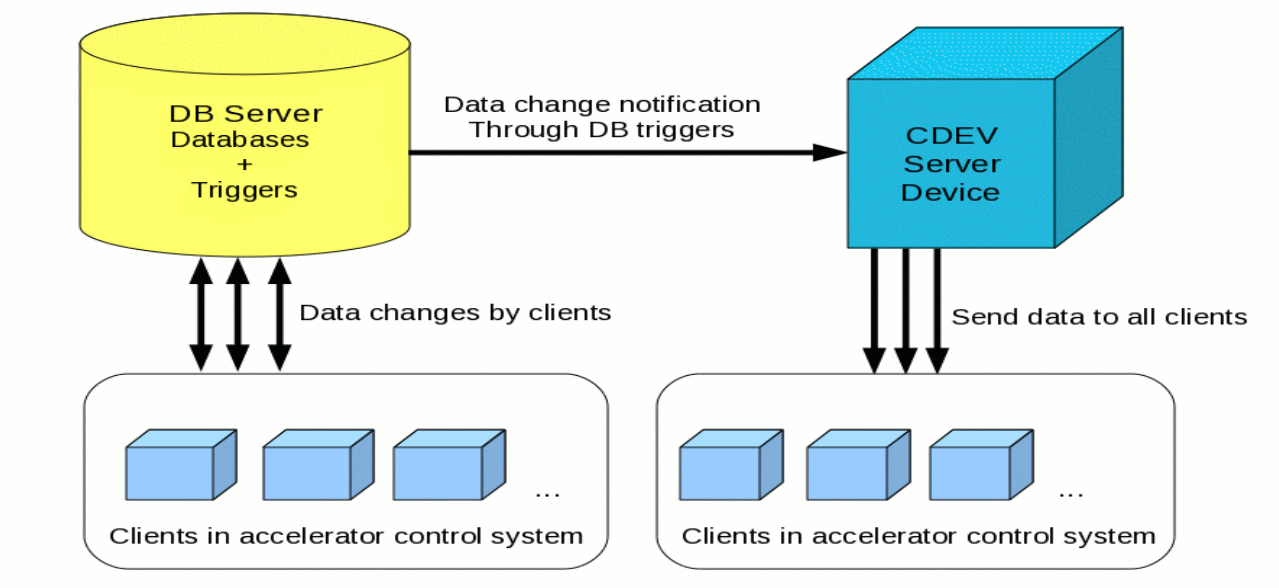


Figure 1: Diagram of ADCN system setup.

1. A database which stores the controls data. The database can exist on any DBMS system which supports the database trigger feature and able to execute system commands from the database trigger. Some DBMS examples are Oracle, MS SQL Server, Sybase ASE, and MySQL. In a target database, a specially designed trigger can be set up to fire whenever the target control data changes. The data changes can be in the form of newly inserted data, updated data, or removal of existing data. Depending on the capability of the DBMS system, the trigger can fire before or after the data changes in the database. The database trigger can also be designed to determine what kind of DCN message to send. The developer set the system up so that it sends events of data changes, meta data of the changed data, or the changed data itself. In this case the trigger will send the DCN message immediately along with the data change that occurred in the database.
2. A second requirement is for the existence of a generic server that listens for the trigger and transmits the DCN message from the database to interested clients. The generic server here can be any well established accelerator control system server such as an ADO or CDEV Server. In the example above CDEV server devices are used.
3. The third requirement is for a client to set up the communication exchange with the server. This allows the client get notified when the target data changes initiated by other clients. The data change process between client and server is independent from these clients of being notified. At the time of a DCN, the data within the database table is updated which initiates the asynchronous transaction to interested clients. In addition to being a place for control data storage, the database becomes integrated into the accelerator control system.

The key component of this system is the design of the database trigger. This is the only part that requires knowledge of SQL programming. The trigger requires the user to determine the type of data to send to clients and under what conditions. Depending on the DBMS system used, the execution of a trigger may or may not affect the database operation in the case where the trigger execution fails. It is good practice for the developer to make sure the trigger has the flexibility to support various failure modes.

In the ADCN system shown in Figure 1, the set up of the CDEV server device is straight forward due to the developers familiarity with well established CDEV (ADO) interface. The CDEV server device serves as a reflective device (the server only needs to forward data from the database server to interested clients). The server supports the same API (i.e. SET/GET methods) used by other devices in the accelerator control system. In addition to specialized applications that use the ADCN system in the Collider-Accelerator Controls System at Brookhaven National Laboratory, generic wide use applications such as GPM (Generic Parameter Monitor), pet (Parameter Edit Tool), and the logging system can also take advantage of ADCN.

ADCN SYSTEMS USED IN RHIC

In the RHIC-AGS controls system, several ADCN systems have been set up by using the methods described in this paper and have been used operationally over the past several years. The DBMS systems involved in these ADCN systems are MS SQL Server 2005 running on a Window PC and Sybase ASE 15 running on Solaris 10.

The same ADCN set up strategy can be applied to any DBMS system which supports the trigger feature. Below is a brief description of one of the ADCN system--- the RHIC access controls system for live IRIS data communication.

In the RHIC access controls system, IRIS recognition data are generated from the IRIS access control devices and stored in a MS SQL Server 2005 database on a security sub-network on a Windows PC. The MCR (main controls room) of RHIC requires asynchronous notification whenever IRIS data in the database is updated. To achieve this, a database trigger was created in the MSSQL database. The trigger initiates the transfer of the changed data to the pre-defined CDEV server running on a Linux host. The MCR application monitors and logs the data from the CDEV device and uses the live IRIS data change for security access in the CAD complex. Figure 2 shows how the system works.

In this example, the ADCN system seamlessly integrates the IRIS access control system (manufactured by LG) and MS SQL Server (2005) database with the in-house designed access control system.

SUMMARY

Asynchronous data change notification (ADCN) between database server and clients can be realized by combining the use of a database trigger mechanism, which is supported by major DBMS systems, with server processes that use client/server software architectures that are familiar in the accelerator controls community (such as EPICS, CDEV or ADO). This approach makes the ADCN system easy to set up and integrate into an accelerator controls system. Several ADCN systems have been set up and used in the RHIC-AGS controls system.

REFERENCES

[1] Oracle@ Database New Feature Guide: 11g Release 1(11.1), B28279-03, October 2008.
[2] Microsoft SQL Server: http://msdn.microsoft.com/en-us/library/ms130214%28v=sql.100%29.aspx
[3] BM DB2: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp
[4] Sybase: http://infocenter.sybase.com/
[5] D.S. Barton et al. RHIC Control System, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* Volume 499, Issues 2-3, 2003, Pages 356-371.
[6] CDEV:http://www-csr.bessy.de/control/Soft/cdev/cdevIntroduction.html
[7] EPICS: http://www.aps.anl.gov/epics/

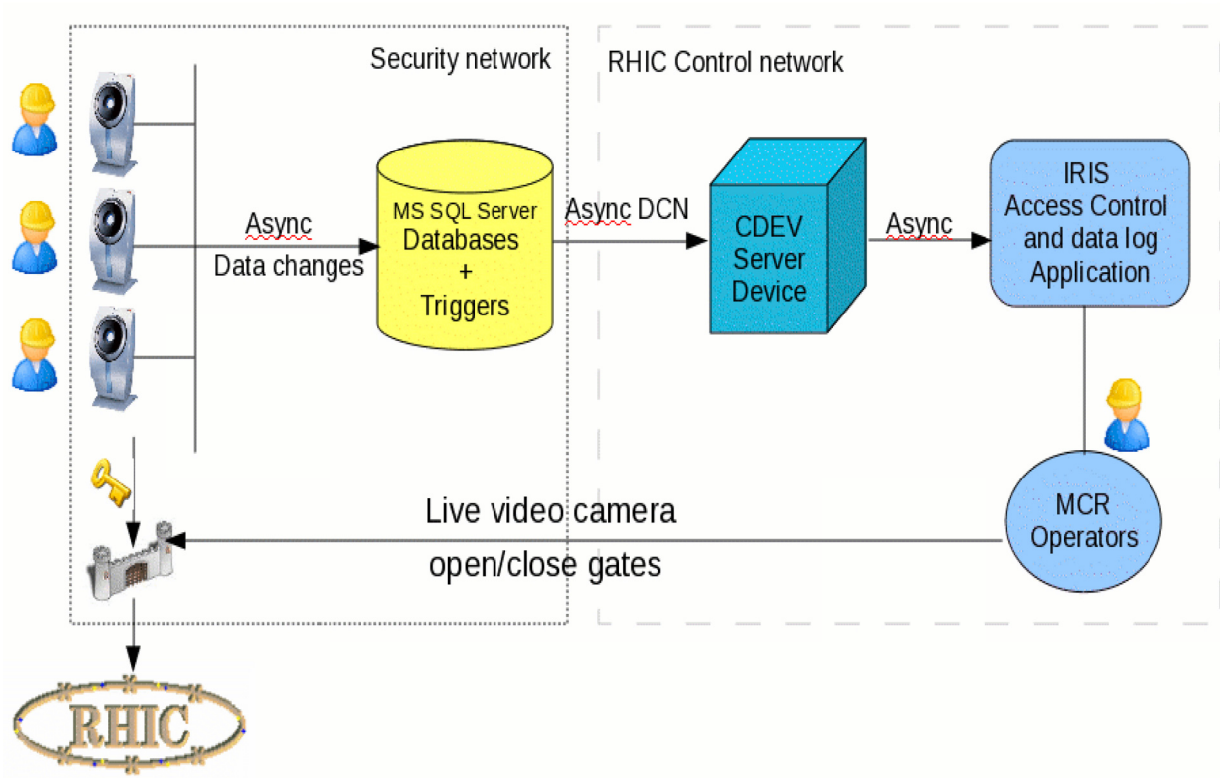


Figure 2: Diagram of RHIC IRIS Access Control System with ADCN.

THE INTEGRATION OF THE LHC CRYOGENICS CONTROL SYSTEM DATA INTO THE CERN LAYOUT DATABASE

E. Fortescue-Beck, R. Billen, P. Gomes, CERN, Geneva, Switzerland

Abstract

The Large Hadron Collider's Cryogenic Control System makes extensive use of several databases to manage data appertaining to over 34,000 cryogenic instrumentation channels. This data is essential for populating the software of the PLCs which are responsible for maintaining the LHC at the appropriate temperature.

In order to reduce the number of data sources and the overall complexity of the system, the databases have been rationalised and the automatic tool, that extracts data for the control software, has been simplified. This paper describes the main improvements that have been made and considers the success of the project.

INTRODUCTION

The LHC Machine

The **LHC Machine** is a 27 km hadron collider, laying 100 m underground, and comprising eight sectors; each one is made of 2 long straight sections and an arc, with 23 regular cells of 107 m in a continuous cryostat.

The Controls Architecture

In each sector, the cryogenic instrumentation is controlled by two Siemens-S7® Programmable Logic Controllers (PLCs) [1, 2].

The man-machine interface for the operators is based on a SCADA (Supervisory Control And Data Acquisition) built using the commercial PVSS® software package developed by ETM, a Siemens subsidiary.

CIET (Cryogenics Instrumentation Expert Tool) is a dedicated SCADA tool used by the cryogenic instrumentation experts.

All cryogenics software (SCADAs and PLC software) conform to the CERN UNICOS (Unified Industrial Control System) framework [3] and is automatically produced by a UNICOS generator. The UNICOS generator takes as an input a specification file containing a list of instruments and parameters. The main aim of this project is to generate this file directly from the databases.

The Databases

Since 2005 the Cryogenics Group has been intensively using several databases (DBs) to manage data appertaining to 34,000 instrumentation channels.

The **Layout DB** is the principal database for centrally maintaining the topology of all CERN installations [4].

Thermbase is a dedicated DB containing the calibration data for all thermometers in the LHC. Its successor **Sensorbase**, is under development and aims to consistently manage all of the metrological data for pressure sensors, level gauges and heaters, in addition to thermometers.

The **MTF** (Manufacturing and Test Folder) **DB**, holds information about individual pieces of equipment. This includes properties, test measurements, status flags, and strict follow-up of the steps of manufacturing, assembly & test procedures.

The **Controls Layout DB** acts as an interface, combining the data from Layout, Thermbase, Sensorbase and MTF into database views which provide the complete set of information used to automatically generate the specifications for the control system.

WHY USE THE LAYOUT DATABASE FOR CRYOGENIC INSTRUMENTATION

The Layout database was initially developed in 2003 for planning the installation and commissioning of the components in the LHC. It aims to capture the system architecture and the details of the installed components in the accelerator complex. Therefore most of the Groups who are responsible for accelerator equipment describe their architecture in this database. Figure 1 shows a few examples of domains currently using Layout; however in principle any domain can be described.

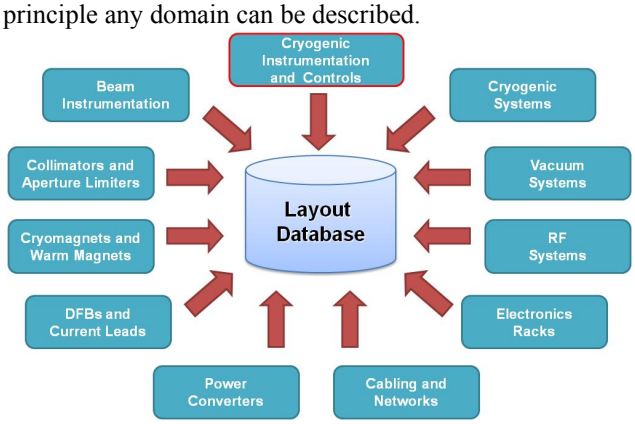


Figure 1: Domains which use the Layout Database.

Unexploitable Data

In 2005, the Cryogenic Instrumentation and Control Section needed to structure a large amount of operational data in a database so that it could be used to define the configuration of the front-end crates, for manufacturing, and also to produce the specifications for the control system.

The data describes the instrumentation attached to magnet assemblies and cryogenic distribution, as well as the electrical components of the controls infrastructure including the cables, connections, pin-outs, electronic modules, crate and racks. Up until this point it was being stored in spreadsheets, plans, drawings and technical

documents – usually on team members' local hard disks. Consequently, the data was dispersed, duplicated, incomplete and inconsistent.

This data forms the basis for nearly all of the key tasks performed by the instrumentation team, but without consolidating it in a database, it would be impossible to extract and share accurate, useful information.

Centralising and Consolidating across Domains

By deciding to integrate the majority of their data into the Layout database, the cryogenics engineers benefitted from having just one single, reliable source of information which was accessible to all team members.

Furthermore, due to the centralised nature of the Layout DB, they profit from the global integration with data from other systems, such as PLCs, field buses, magnets, cryogenics distribution, electrical circuits, etc. This integration means that maintenance across domains is easier to manage and greatly helps the machine operators to diagnose problems more rapidly.

Data Accessibility

The existing Layout web interface can be used to browse the data, easily traversing the data structure and the relationships between objects through hyperlinks. Also, since Layout is fully integrated with MTF, it is possible to directly access, via the web interface, all of the manufacturing and test data concerning the specific piece of equipment which is installed in a particular Layout position. This integration with MTF facilitates traceability, as the database can track over time all of the locations in which a component has been installed and also all repairs it has undergone. This is essential information required by international regulations on nuclear safety.

The seamless integration between UNICOS, Layout and MTF allows direct access from any instrument in SCADA to its functional information on the Layout web interface, and then to its individual properties and history in MTF.

Other useful inbuilt features of the Layout DB include versioning and the ability to track changes to the data.

EVOLVING FROM A FIRST APPROACH

Initially, only the topology of the LHC's physical instrumentation was recorded, with a limited number of object types.

Hence, conceptual objects required for the control system were not catered for in the Layout DB structure. These objects included 'instruments' which do not physically exist but are derived from others, like calculated flow or max/min values; and variables not easily related to anything physical, like spare objects or status information exchanged between PLCs.

Therefore tailor-made database structures such as tables and procedures were implemented in a supplementary Controls Layout database in order to either store or dynamically generate these new types of objects.

Rationalising the Physical and Conceptual Channels

Over time, the scope of the original Layout data model was broadened as it was acknowledged to be flexible enough to accommodate a wider range of objects and properties. Thus, it became possible to apply a coherent treatment to both physical and conceptual objects.

The major benefit of grouping data in the Layout DB and treating all types of instrumentation channels the same way was that the overall data structure became simpler and the data was easier to maintain. In addition, the views for the controls specifications became less complex, as it was no longer necessary to maintain distinct pieces of code to separately retrieve each category of instrument.

The Need for Inheritance and Relationships

The Layout DB supports inheritance. This means that derived instrumentation channels could automatically inherit properties from their parent, thus eliminating inconsistencies which could have been introduced if modifications to the parent instrument were not explicitly reapplied to the child.

As well as implementing conceptual objects in Layout, new types of conceptual relationships between instruments have also been defined; for example the link between a virtual flow meter and the instruments used in the flow calculation, such as its upstream thermometer or valve aperture, as illustrated in Figure 2.

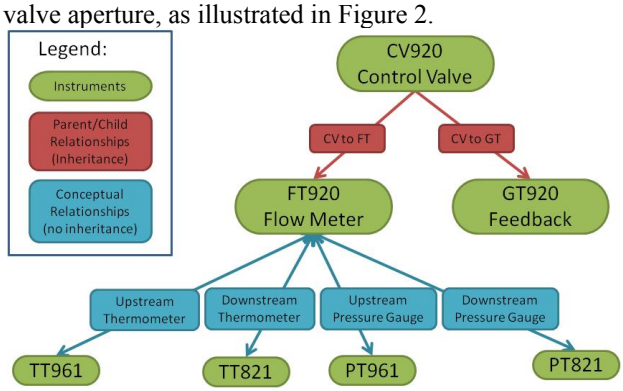


Figure 2: Inheritance and conceptual relationships.

The Introduction of Systems

Up until this point the emphasis had been focused on implementing the individual IOs in the Layout database. However there were other software objects required for the control system such as Process Control Objects (PCO), Controllers, Alarms and Interlocks, which are high-level relationships between sensors and actuators. These objects were successfully integrated into the Layout DB by defining them as "Systems", which are groups or hierarchies of Layout objects.

Figure 3 shows how high-level PCO systems are defined as a hierarchy of lower-level PCOs, which in turn are hierarchies of controller systems. At the controller level, the systems link to the IOs, as a controller is specified as a group of instrumentation channels.

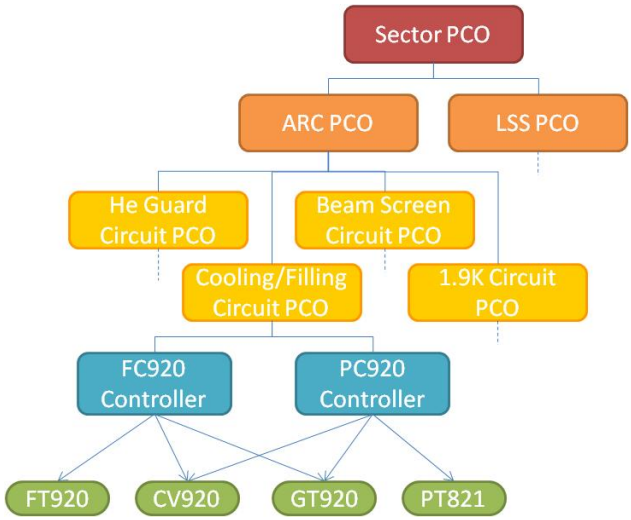


Figure 3: a hierarchy of systems.

COMPLETING THE DATA

After defining the structures of all types of instrumentation channels and control loops in the Layout DB, the data could be completed and all other data sources eliminated. Before this parametric data was integrated into the Layout DB it was thoroughly checked for consistency.

Importance of Data Quality

In the LHC, Instrument functionality is characterised by a dedicated attribute called Tag name. This implies that all instruments with the same Tag name should have similar values for Range, Scale, Deadband and Format due to having the same typical operation characteristics. For example, PT821 pressure sensors found in the magnet cold masses have a typical Range of 0 to 20bar. Whereas the PT891 pressure sensors on the DFB (deltaP line) have an operational range between -350 and +350mbar.

The SCADA display also depends on the Tag name, in harmony with the measurement range. For example, the reading from a PT821 sensor is formatted on the SCADA display as 2 integers plus 2 decimals, whereas a PT891 sensor is displayed as 3 integers. All of this information is stored in the Layout DB and propagated to the control system for use by the operators.

Maintaining Data Consistency

The original values of parameters such as Range, Scale, Deadband and Format were defined before the LHC was operational and were often estimates. Since the machine had been operational for nearly a year, many of these parameters had been manually adjusted in PVSS by experts and operators in order to improve the performance of the control system. Storing these values in the database safeguards these modifications when the control system software is regenerated.

Therefore, for each distinct Tag name present in the Layout DB, the set of current operational parameters were extracted from the SCADA. For those Tag names having conflicting parameter values, the discrepancy was resolved in conjunction with the cryogenics experts.

It was important to review these parameters because of the significant impact they have on the performance of the control system. For example, the Deadband value defines the type of filtering performed by the SCADA archiving. By default PVSS logs data at a rate of 1Hz. Filtering is required to reduce the raw data volumes by eliminating noise and keeping only relevant data long-term.

Although these corrected parameters were grouped by Tag name, they were propagated to the Layout DB as properties of individual channels. This allows future adjustments at channel level if an exception to the general rule is observed during operation.

The Description as a Key Attribute

"Description" is a very important attribute as it indicates the function of an instrument and therefore facilitates the work of the operator. In the SCADA, the description shown on the instrument panel includes the instrument functionality concatenated with field bus address, (the source of both of these pieces of data is the Layout database). This makes it easier for maintenance people to know where to find the instrument in the field bus without having to refer to other data sources.

Before the descriptions were reviewed, some were missing, incorrect or incoherent between instruments with the same tag name or similar functions. All descriptions were rechecked so that they were consistent in content and format across all types of instrument and then updated in the Layout Database.

GENERATING SPECIFICATION FILES

Business Logic in the Application

Until 2010, the control system specifications were produced by an automatic generator that extracted data from several DB views and various external files and applied a complex set of rules and calculations, to derive parameter values, relationships and secondary objects.

This Specification Generator was a very complex application which underwent a great deal of re-patching. It contained an enormous amount of code to handle each special case and exception; and thus it became difficult and time-consuming to maintain. With the rationalisation of the databases, it rapidly became unmanageable.

Transferring the Business Logic to the DB

It was critical to remove all of the knowledge embedded in the generator code and transfer it to the database. However the logic was not simply recoded. Instead, the results given by the rules were imported directly into the database as data items. This could be done because by 2010 the LHC cryogenics configuration was reasonably stable and the set of values calculated by the generator was complete. This meant that it was no longer necessary to automatically recalculate the values. Any future corrections for individual exceptions can be

manually implemented; if the original rule changes, the data can be re-imported as a batch.

The main advantage of this approach is that all of the data in the specification files is available in the database, as either persistent or aggregated values. By converting generation rules into data the amount of calculated data and information hidden in the code has been minimized. Consequently, other applications can use this data without having to re-implement the logic. Also, the code in both the generator and the database is simplified, greatly reduced and maintainable.

Simplifying the Specifications Generator

Once all of the data was coherently structured in the database, a set of views were produced which exactly replicate the structure of each page of the IO specifications. These views are stored in the Controls Layout database which acts as an interface for assembling the data from Layout, Sensorbase and MTF.

The Specifications generator has been replaced by a simple application which extracts all of the required data from each view, for a given sector and PLC. It formats the data as either a text file – in the case of the Hardware Configuration Specifications – or as an Excel file.

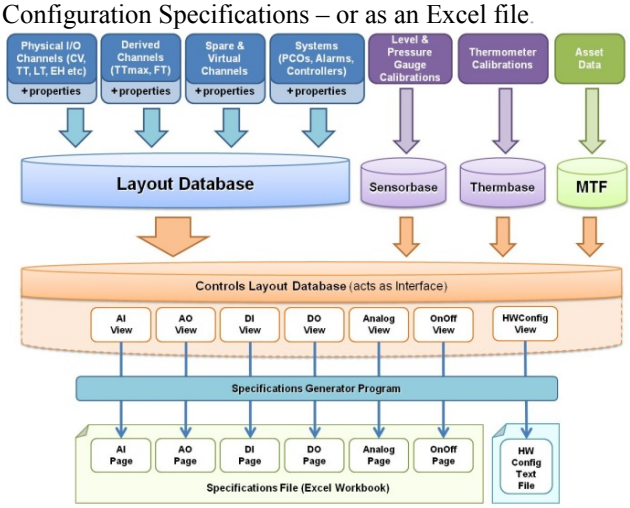


Figure 4: Generating Specifications from the Databases.

It is important to note that there is no business logic in the new generator application and it does not extract any data from external files.

FUTURE IMPROVEMENTS

Aside from on-going maintenance the cryogenics instrumentation data is currently stable.

New Client Requirements

With the upgrade of the UNICOS programming environment to version 6 in 2012, the format of the specifications files will be modified [5]. Currently, the same specification files are used to generate both the Operator's SCADA (based on UNICOS) and the Experts SCADA (CIET), but in the future two separate sets of specification files will be required. This will not imply a change in the data or the underlying database structure, but new database views corresponding to the new file formats will need to be developed. In addition, the generated files should be in xml format which implies a change to the template integrated in the new generator program.

Remodelling of the Layout Database

During 2011 the Layout database will undergo a significant restructuration. However, the specifications views should remain largely unaffected because they extract the majority of their data from an all-inclusive summary view. This complex view will have to be completely rebuilt in order to retrieve the instrumentation data from the new database structures in a way that is transparent for the specifications views.

Improved Access and Availability

A major new initiative planned for 2012 is the development of a read/write interface for the Layout Database. It will combine the browsing functionality of the current Layout web interface with writing capabilities which will allow the equipment owners to introduce modifications, making them fully responsible for their own data. Procedures for maintaining data quality will be established and integrated into the tools.

CONCLUSIONS

We have achieved our primary goal of making databases the only data source required for generating the specifications for the cryogenic control system.

Throughout this project we have strived for an excellent standard of data quality, as it is a fundamental requirement for achieving a fully data-driven system.

By decommissioning the previous Specifications Generator, the databases have become critically important. Therefore, the structure and the data must be meticulously maintained in order to ensure the reliable operation of the LHC cryogenics system.

REFERENCES

[1] P. Gomes et al., "The control system for the cryogenics in the LHC tunnel", ICEC22, Seoul, Korea, Jul-2008, p. 45.

[2] P. Gomes et al., "The Control System for the cryogenics in the LHC tunnel [First Experience and Improvements]", ICALEPCS09, Kobe, Japan Oct-2009.

[3] Ph. Gayet, R. Barillère, "UNICOS a Framework to Build Industry-like Control Systems Principles Methodology", ICALEPCS05, Geneva, Switzerland 2005.

[4] P. Le Roux et al, "The LHC Functional Layout Database as Foundation of the Controls System", ICALEPCS07, Knoxville, USA, Oct-2007, p. 526.

[5] B. Fernandez Adiego et al., "UNICOS CPC6: Automated Code Generation for Process Control Applications", these proceedings.

INTEGRATING THE EPICS IOC LOG INTO THE CSS MESSAGE LOG*

K.U. Kasemir, E. Danilova, ORNL, Oak Ridge, TN 37831, U.S.A.

Abstract

The Experimental Physics and Industrial Control System (EPICS) includes the "IOCLogServer", a tool that logs error messages from front-end computers (Input/Output Controllers, IOCs) into a set of text files. Control System Studio (CSS) includes a distributed message logging system with relational database persistence and various log analysis tools. We implemented a log server that forwards IOC messages to the CSS log database, allowing several ways of monitoring and analyzing the IOC error messages.

EPICS IOC LOG

The Experimental Physics and Industrial Control System (EPICS) front-end computers, commonly referred to as Input/Output Controllers (IOCs), supports a simple TCP-based logging system. IOCs connect to the "IOCLogServer", by default on TCP port 6500, and send basic text messages. A new line character terminates each text message. The IOCLogServer annotates each message with the current time/date and the IP address of the IOC, and then writes them to a rotating set of text files [1].

This logging mechanism has proven to be very robust, but it lacks tools to analyze the log messages. Use of the IOC log at the Spallation Neutron Source (SNS) was limited to a post-mortem search for clues in case of IOC problems, based on UNIX command-line text tools like "grep" or directly reading the log files in a text editor.

CSS LOG

CSS applications utilize a Java Message Server (JMS) to issue log messages [2]. The same mechanism is also used for network traffic between components of the CSS alarm system [3]. All CSS JMS messages can be routed to a relational database (RDB) for later analysis.

Fundamentally, each JMS message can consist of arbitrary property/value pairs, but the following standard properties should be included to allow analysis of messages from different sources:

- TYPE: A value of 'log' indicates a log message.
- DATUM: Date and time of the message.
- TEXT: For log messages, this is the actual message text.
- SEVERITY: An indicator for the message severity, for example "INFO"
- HOST: The name or IP address of the computer that sent the message.
- USER: Name of the user who submitted the message.
- APPLICATION-ID: Name of the application that submitted the message.

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy

Fig. 1 shows the generic CSS Message Log Viewer that can display selected properties of messages that have been logged to the RDB. This viewer also allows filtering, for example to restrict the display to messages of a certain TYPE, messages that originate from a specific HOST, or messages with TEXT that contains a certain word. This viewer allows interactive review of messages.

Since the messages are stored in a relational database, the information is accessible to a wide variety of tools. It is for example relatively easy to create reports that a re targeted to the analysis of a specific problem, as we will describe in the section called JSP Message Reports.

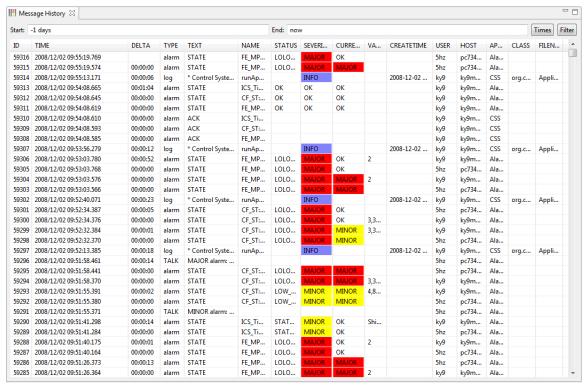


Figure 1: CSS Message History Browser.

IOC LOG SERVER REPLACEMENT

To benefit from the existing infrastructure for CSS message logging, we implemented a new IOC Log Server in Java that operates similar to the original EPICS IOCLogServer. It also listens on TCP port 6500 for text messages, but each received line of text is written to the CSS message RDB with the following properties:

- TYPE: A value 'iocLog' identifies IOC messages.
- CREATETIME: Data and time when message was received.
- TEXT: Original message text.
- HOST: IP address of the network client, i.e. the IOC.
- SEVERITY: Defaults to "INFO"
- APPLICATION-ID: "IOCLogServer"

To EPICS IOCs, the replacement of the original log server for text files with the new implementation is transparent. We simply start either the old or the new log server, and IOCs will use the current instance as soon as they attempt to log a message.

Filtering

We noticed early on that the IOC log messages were dominated by repeated messages. Certain IOCs would

often send the exact same message every few seconds. To reduce the amount of space used by such messages in the RDB, we added a simple filter:

If the same network client (i.e. IOC) sends the same message text, this is not logged as a separate message. Instead, a repeat count is updated in the REPEATED property of the original message.

The filtering is based on the last message received from each network client, only comparing a new message with the last message from that client. If a client keeps sending a message with text "A", this will be filtered by the repeat count mechanism. If a client should repeat two messages in a pattern "A", "B", "A", "B", …, this will not be filtered out. While we do observe this pattern, it has at this time not been frequent enough to warrant checking all recent messages of a client for repetitions.

In addition to filtering repeated messages, there is a pattern-based filter. Based on regular expressions for the message content, it can be configured to ignore messages that match a pattern. This can for example be used to suppress known benign status messages that we prefer to exclude from the log.

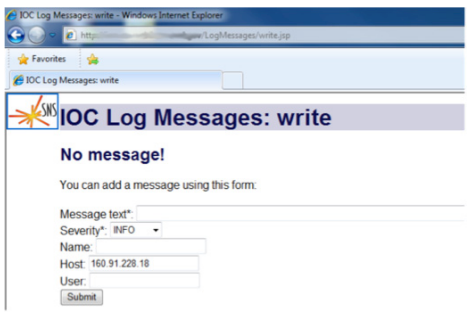


Figure 2: Web form for submitting a message.

JSP LOG WRITER

The IOC Log Server replacement is convenient for EPICS IOCs, but less accessible to control system front-end computers that are implemented by other means.

By wrapping the essential Java code for logging messages to the RDB as a Java Server page (JSP) for Tomcat [4], we created a web service that can for example be used by programs implemented in National Instruments LabVIEW to submit log messages.

The web service is accessible via a URL similar to the following for submission of a basic text message:

http://server/LogMessages/write.jsp?TEXT=…

If desired, additional URL parameters like SEVERITY or USER can be provided to set the associated message properties.

If no message parameters are included in the URL, a web form as shown in Fig. 2 is presented to allow interactive entry of the message content.

JSP MESSAGE REPORTS

With all IOC log messages available in the relational database, we are now able to create various online reports. Fig. 3 shows a Java Server Page (JSP) that summarizes recent IOC log messages.

A pie chart depicts the "Top 10" IOCs that produced the highest number of messages in the analyzed time period, which defaults to the last hour. Tables list repeated messages as well as all messages of the time period. Both tables contain the message text but also time, severity, host, number of repeats, and optional user name. By default, the tables are sorted by the time when each message was created, but users can sort the tables by any column by clicking on the table column header.

A web form allows users to configure the time range, provide pattern for the IOC name, and choose the number of messages to display.

A detailed list of messages for a particular IOC is accessible by either clicking on the associated sector on the pie chart sector (if this IOC is one of top 10 message producers), or by selecting the IOC name from drop-down lists of IOCs that are sorted by name as well as message count.

These JSP reports are available to any computer with a web browser at our site.

SUMMARY

Without any change to EPICS IOCs, we were able to transparently route IOC log messages from the original file-based system to the CSS message log that is stored in a relational database. Web reports show for example which IOC produced the most messages in the last few days.

In the past, our IOC log files were only consulted "after the fact" when an IOC had run into severe problems. Now we are able to spot an increase in message traffic, hopefully anticipating a more severe error.

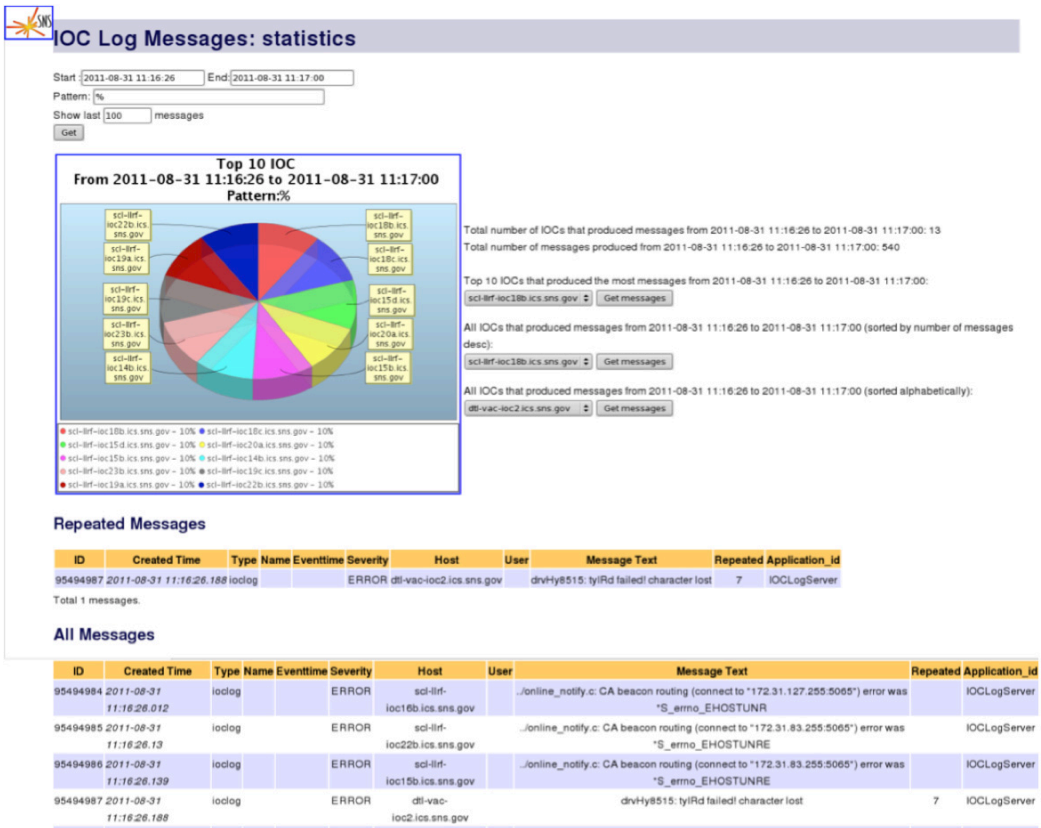


Figure 3: JSP Report of IOC Log Message statistics.

REFERENCES

[1] M. Kraimer, J. Anderson et al, "EPICS Application Developer's Guide, Release 3.14.12", http://aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide.pdf.

[2] K. Kasemir, G. Carcassi, "Control System Studio Guide", http://cs-studio.sourceforge.net/docbook

[3] K. Kasemir, "The Best Ever Alarm System Toolkit", ICALEPCS 2009, Kobe, Japan

[4] Tomcat web application server, http://tomcat.apache.org

BDNLS - BESSY DEVICE NAME LOCATION SERVICE

D. Engel, P. Laux, R. Müller, HZB, Berlin, Germany

Abstract

Initially the relational database (RDB) for control system configuration at BESSY has been built around the device concept [1]. Maintenance and consistency issues as well as complexity of scripts generating the configuration data, triggered the development of a novel, generic RDB structure based on hierarchies of named nodes with attribute/value pairs [2]. Unfortunately, it turned out that usability of this generic RDB structure for a comprehensive configuration management relies on sophisticated data maintenance tools. On this background BDNLS, a new database management tool, is currently under development using the framework of the Eclipse Rich Client Platform. It uses the Model View Controller (MVC) layer of JFace to cleanly separate retrieval processes, data path, data visualization and actualization. It is based on extensible configurations defined in XML allowing to chain SQL calls and compose profiles for various cases. It solves the problem of data key forwarding to the subsequent SQL statement. BDNLS has the potential to map various levels of complexity into the XML configurations. This provides usable, tailored database access to configuration maintainers for different underlying database structures. Based on Eclipse, the integration of BDNLS into Control System Studio [3] is straight forward.

HISTORY OF CONCEPT

Views – The First Approach

Database-views are predefined SQL-queries. They are like macros with precached results. Views are ideal for frequently used large and complex queries typically used in small database-applications. A big drawback of views is, that they are not always able to update their data sources. Views are able to hide the complexity of data structures, but you have to create one view for one specific problem. If you need to create views for all aspects of a facility, then you need hundreds of them. Complexity of data structures is pseudo-simplified in a confusing amount of views.

Generalization of the Data Structure

The obvious idea to maintain full flexibility, was to separate the data sets to the maximum, like atomic elements. For that purpose, the whole data set, for example device properties or I/O specifics, has to be to modeled in data and relations. The data is not grouped by table structures or specific column names. Only links and relations connect the data. This is a clean method to retain the data and avoid redundancy. This data structure has been called Gadgets [4]. This solution has the drawback, that you manage a very complex structure of stored data. Visualizing and managing the data is problematic because it is abstract and queries on this structure are very complex. It is not a general solution for storing data.

The Experiences from a Prototype

The first graphical application was implemented using Eclipse as a Rich Client Platform (RCP) application. RCP provides many solutions of graphical interfaces and a plugin concept to assemble numerous small plugins to a full-featured application. The application was designed as well as a standalone Rich Client Application and as plugin for other Rich Client Applications like the Control System Studio (CSS) [3] Framework. The application is able to provide the location of a device and to view their properties as requested. This solution used static SQL-queries and a special view that provided all information for a complete navigation tree, broken into columns. Updates of the data is only possible with hard coded update-dialogs. Another drawback of this approach was, that data could only be updated using the dialogs provided.

The Generalized Approach

The first graphical approach revealed with following shortcomings:

- Search, read, change and insert elements in the database without knowledge of SQL by the user.
- Importing and exporting huge data sets.
- The profile contains the navigation chain to a target element, the element properties, the wizards[1] and the popup-menus available for this target.
- The profiles should be able to change during operation.
- The complexity of data structures and SQL-queries is extracted from the source code and encapsulated to a configuration file. The source code is generic.
- The application should be independent of the given data structure. You can design your own SQL-queries in the configuration file.
- With the use of Java, a platform independent application is provided.
- The application uses existing databases and no hard coded data structures. It is also not designed to follow all data relations given by data keys hence it is no competitor of IRMIS [5].

[1] A Wizard is a user interface of dialogs to help the user to manage complex inputs

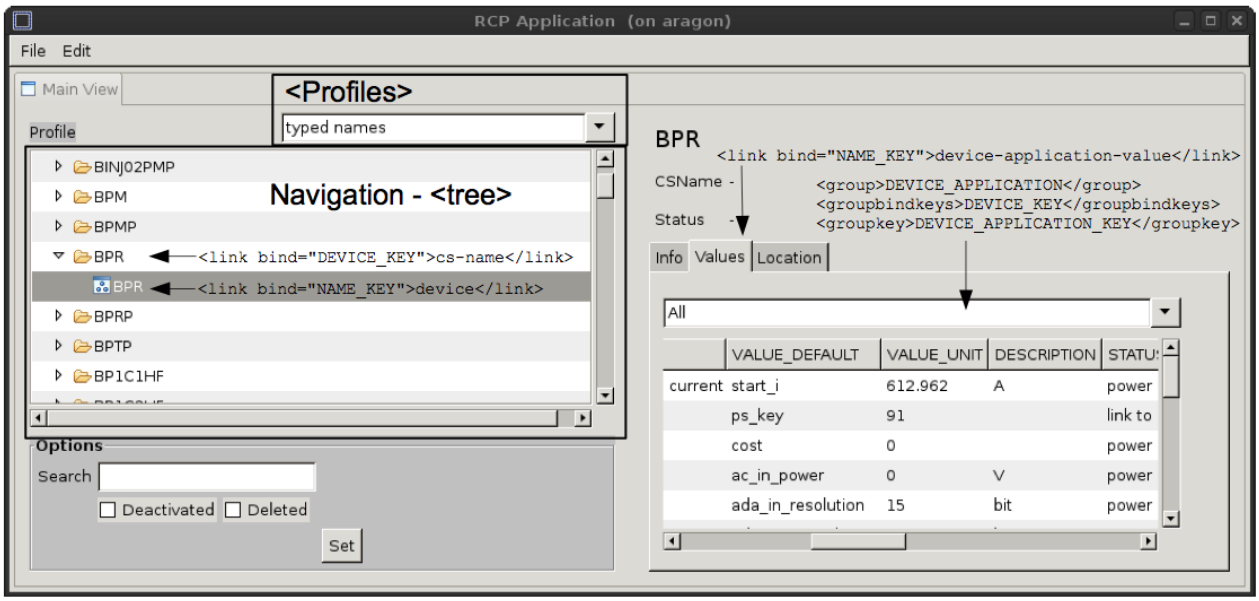


Figure 1: BDNLS prototype application; Profile selector (top), tree navigation (left), view section (right), the query section not visible.

XML CONFIGURATION CONCEPT

The whole configuration of the BDNLS-application is defined in an XML-file. The configuration file contains:

- How to access the database (URL, user name and the password).
- Definition of various profiles allowing several SQL-query sequences and different views.
- The displayed element and the data key provided by the SQL-Query.
- The definition of the navigation tree (-by profile-) and the data keys required for next tree element.
- The provided properties of a selected device, arrangement, grouping and visualization settings.
- The popup-menus and wizards provided for a device.

Profiles

A profile defines the navigation to a device and its specific properties. profile names are declared in the `<profile>`-part. The default profile used when the application starts.

```
<profiles default="1">
   <profile>located names</profile>
   <profile>typed names</profile>
   <profile>typed devices</profile>
   <profile>by facility and family</profile>
</profiles>
```

Navigation

The navigation section is divided in to different profiles. Every profile consists of the parts `<tree>` (shown as a tree elements) and `<tab>` (shown as tab elements). The tree-part is used to display and to navigate through device classes, domains, families and facilities. `<link>`-parts are called sequentially when expanding a tree element. `<link bind="xxx">` is used to narrow the results at the next hierarchy, using the keys from the last results. Clicking at the last element in the tree-part, initiates processing of the tab-part.

```
<navigation>
<profile name="located names">
   <tree>
      <link>facility</link>
      <link bind="FACILITY_KEY">cs-domain</link>
      <link bind="NAME_DOMAIN_KEY">cs-subdomain</link>
      <link bind="NAME_SUBDOMAIN_KEY">cs-name</link>
    </tree>
    <tab>
       <link bind="NAME_KEY">device-info</link>
       <link bind="NAME_KEY">device-location</link>
       <link bind="NAME_KEY">device--value</link>
    </tab>
</profile>
</navigation>
```

Query Section

The query section describes the name of the query, the corresponding statement `<statement>`, the provided key names `<key>` of the query and the column name to be shown as the element name`<label>`. The provided keys are stored and can be used to narrow the results of the next query. A statement may yield zero, one or many keys, they are separated by a comma.

```
<link name="cs-subdomain">
    <statement>NAME_SUBDOMAIN_DROPDOWN</statement>
    <key>NAME_SUBDOMAIN_KEY</key>
    <label>NAME</label>
</link>
```

View Section

Currently, the view section is only used for the properties of a device. The view part defines the name of

a tab <title>, whatether it shown as a form or a list <showtype> and which type of context menus the elements provides <showmenu>. Additionally, the output can be filtered by a group query. To group/filter the device properties, you have to define a group statement <group> and a grouping/filter key that is used to restrict the results of the device properties. To restrict the group statement itself, you can restrict the statement with another key provided by the device properties.<groupbindkeys> The tag <showmenu> links to the required popup-menus.

```
<view>
<tab name="device-value">
    <title>Values</title>
    <group>DEVICE_APPLICATION</group>
    <groupbindkeys>DEVICE_KEY</groupbindkeys>
    <groupkey>DEVICE_APPLICATION_KEY</groupkey>
    <showtype>List</showtype>
    <showmenu>device-value</showmenu>
</tab>
</view>
```

Popup and Wizard Block

Manipulation of the data, is enabled by an update or input wizard defined within the popup tag <popup>. The key expected by the wizard is defined with the tag <column>.

```
<popup name="device-value">
    <item name="create">
       <wizard>wizard-device-value</wizard>
       <column>device_key</column>
    </item>
    <item name="edit">
       <editor>editor-device-value</editor>
       <column>device_value_key</column>
    </item>
    <item name="remember">
       <history>history-device-value</history>
       <column>device_value_key</column>
    </item>
</popup>
```

IMPLEMENTATION

The Eclipse Rich Client Platform was chosen for implementation. By using and providing plugin extension points, it is possible to use other Eclipse based or third party plugins to extend the application. To connect to the database, a JDBC interface is used. The connection class can load many kinds of JDBC bridges dynamically, like PostgreSQL, MySQL, Oracle and many more. But at the moment only a Oracle OJDBC bridge is integrated in this project. The GUI of the first implementation of this project is shown in Figure 1.

CSS Integration

Adding the plugin-activator class allows to integrate this application as a plugin in CSS. Hence it is possible to exchange EPICS [6] process variables (PV) with CSS, to retrieve archived data, or to get all device information depending on a PV.

FUTURE DEVELOPMENT

- Implementation of drag & drop methods to improve the data exchange with CSS, other windows/forms and applications.
- Integration of visualization function, for example to show the location of a device.
- Integration of EPICS-PVs for fast visualization of the device state.
- Abstraction of the wizards to be fully configurable and provide plugin extension points for custom forms and visualization classes.
- Methods providing import and export formats for CSV, XML and XLS.
- Provide more JDBC Interfaces.

REFERENCES

[1] T. Birke et al., “Relational Database for Controls Configuration Management”, IADBG Workshop 2001, San Jose, CA, USA.

[2] T. Birke et al., “Beyond Devices – An improved RDB Data-model for Configuration Management”, PO1.078-7, ICALEPCS’05, Geneva, Switzerland.

[3] Jan Hatje, M. Clausen et al., “Control System Studio (CSS)”, MOPB03, ICALEPCS’07, Knoxville, Tennessee, USA, http://cs-studio.sourceforge.net.

[4] T. Birke et al., “Introducing I/O Channels into the Device Database Open new Potentialities for Configuration Management”, WEDT003, ICALEPCS’01, San Jose, CA, USA.

[5] D.A. Dohan, “The IRMIS Universal Component-Type Model”, TOPA03, ICALEPCS’07, Knoxville, Tennessee, USA, http://aps.anl.gov/epics/irmis/index.php.

[6] http://www.aps.anl.gov/epics/

DESIGN AND IMPLEMENTATION OF THE CEBAF ELEMENT DATABASE *

T. Larrieu, M. Joyce, C. Slominski, JLAB, Newport News, VA 23606, U.S.A.

Abstract

With the inauguration of the CEBAF Element Database (CED) in Fall 2010, Jefferson Lab computer scientists have taken a step toward the eventual goal of a model-driven accelerator. Once fully populated, the database will be the primary repository of information used for everything from generating lattice decks to booting control computers to building controls screens.

A requirement influencing the CED design is that it provide access to not only present, but also future and past configurations of the accelerator. To accomplish this, an introspective database schema was designed that allows new elements, types, and properties to be defined on-the-fly with no changes to table structure. Used in conjunction with Oracle Workspace Manager, it allows users to query data from any time in the database history with the same tools used to query the present configuration. Users can also check-out workspaces to use as staging areas for upcoming machine configurations.

All Access to the CED is through a well-documented Application Programming Interface (API) that is translated automatically from original C++ source code into native libraries for scripting languages such as perl, php, and TCL making access to the CED easy and ubiquitous.

BACKGROUND

Previous efforts to compile a comprehensive configuration database for the Jefferson Lab CEBAF accelerator and use it to automate the setup or restoration of the electron beam have foundered on the complexity of accommodating the constantly evolving hardware installed in the accelerator and the multiplicity of elements whose properties vary with the increasing energy of each pass through the recirculating LINACS. The progress of the 12GeV upgrade project has, however focused renewed emphasis upon solving these problems by developing a central authoritative configuration database available to existing or future tools needed to operate the revamped accelerator. The resulting product has been coined the CEBAF Element Database (CED).

DATABASE DESIGN

Design

The CED has been implemented using a modified Entity-Attribute-Value with Classes and Relationships (EAV/CR) data model [1]. This design choice is appropriate because the types of information to be stored in the CED will vary from system to system. The database will necessarily evolve over time to support additional configuration attributes for existing systems as well as to model the yet-unknown parameters of the new hardware to be installed for the 12GeV upgrade.

In a traditional database schema, adding support for new accelerator hardware would involve adding additional tables and columns to the database. Such changes impose a burden by requiring software updates before the new schema can be used. In contrast, the EAV/CR data model employed by the CED is introspective – defining a new class of accelerator hardware in the CED simply involves adding rows to the already-existing metadata "catalog" tables (fig. 1). Once defined in the catalog, existing software is fully capable of interacting with the new entities after discovering their properties from the metadata tables.

Versioning

In addition to its flexibility in accommodating future data requirements, the CED schema was also designed to make use of the Oracle Workspace Manager toolkit [2]. Oracle Workspace Manager is used to version-enable the table rows in the CED, and permit simultaneous access to present, proposed, and historical versions of data.

Oracle Workspace Manager accomplishes this feat for a table by renaming it to *table name*_LT, adding a few columns to the table to store versioning metadata, creating a view on the version-enabled table using the original table name and defining INSTEAD OF triggers on the view for SQL DML operations. Clients of the CED do not need to incorporate any special logic in order to interoperate with the different versions stored in the database other than to specify which save point or workspace they wish to access.

Workspaces

A key version-enabling feature of the Oracle Workspace Manager is the ability to compartmentalize a collection of database changes into a logical construct called a workspace.

The primary workspaces is by definition the LIVE workspace and represents the current configuration of the CEBAF accelerator. Additional workspaces may be "checked out" from LIVE and used to stage upcoming configurations or to perform what-if scenarios (fig. 2). The contents of these workspaces can be merged (committed) into LIVE when appropriate.

*Notice: Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177. The U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce this manuscript for U.S. Government purposes.

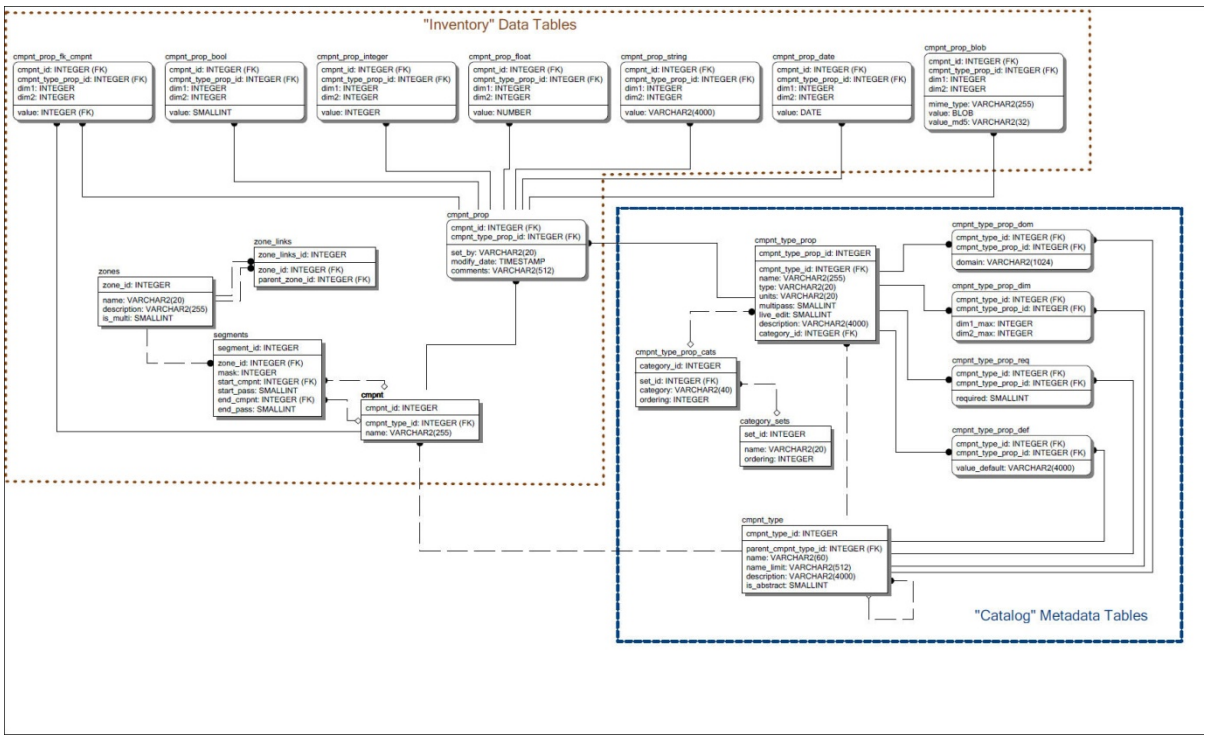


Figure 1: The core CED schema illustrating the "Catalog" tables that store the metadata defining accelerator elements. Element values stored in the "Inventory" tables as strings, floats, integers, etc. are mapped via the metadata into meaningful attributes. New types of elements may be added to the CED on-the-fly by defining their metadata – changes to the database schema are not required.

Save points

Because the Oracle Workspace Manager saves a history of all changes to a row in addition to information about its workspace membership, it is possible to query historical data simply by specifying the date and time as of which the query is being issued. To assist users, so that they don't have to specify exact dates and times, the CED provides named save points for the LIVE workspace corresponding to key configuration milestones such as the beginning or end of an experimental run or of an accelerator maintenance period (fig. 2).

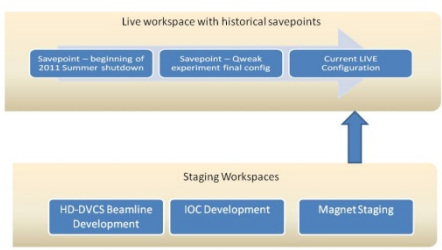


Figure 2: CED Workspaces showing historical save points and staging workspaces used to prepare upcoming changes to LIVE.

In contrast to the typical approach to preserving data history using triggers and a history table containing prior values, where accessing the history requires special one-off queries or custom reports, accessing CED historical data requires no different software and no more effort than accessing the current configuration.

Configuration Control

Workspaces are also used in the CED to effect a level of configuration control. With the exception of a very few properties explicitly identified as live-editable, the CED does not permit users to update data in the LIVE workspace. All updates to the CED must be performed in a staging workspace and audited for validity before being committed to LIVE.

INTERFACE

The generalized nature of the EAV/CR data model employed by CED requires a robust API to interpret the contents of the metadata tables and translate the abstract database storage into recognizable attributes useful to programs and users. The API is also responsible for enforcing much of the data validation and domain logic embodied in the metadata catalog.

For the CED, a shared library was written in C++ to be the sole interlocutor for applications that will access its information. Native versions of the API are available not only to C++ programs, but also to scripts written in Perl, PHP, and Tcl. The script language versions are generated

automatically from the original library via the open source SWIG (Simplified Wrapper and Interface Generator) tool [3,4]. Having the API natively available makes it easier to update the many tools used by JLAB operators that have been written in these scripting languages over the years, as well as to develop new ones. Because they stem from the same source library, all the languages have the same functionality available and benefit from the development effort that went into testing and documenting the original compiled library.

On top of the core API, two general purpose user interfaces to the CED have also been built around the C++ library and its scripting language derivatives: a RESTful web interface and a full-featured command-line tool usable from JLAB Linux workstations.

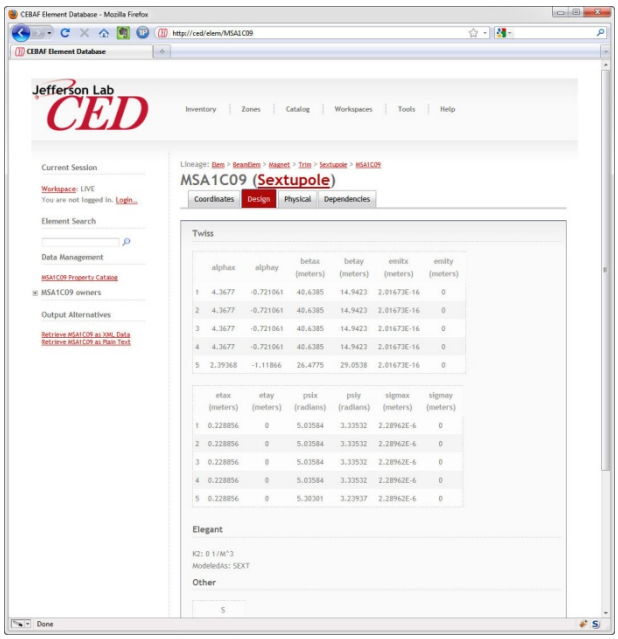


Figure 3: CED Web Interface provides access to CED data in HTML (shown), XML, or plain text format via simple to construct URL patterns.

IMPLEMENTATION

The CED was rolled out to the JLAB community in September 2010 initially populated with data for the Magnet and Input Output Controller (IOC) systems. Data for the RF and BPM systems were added soon thereafter.

In May 2011, CEBAF entered a six months duration shutdown period during which time more than the 200 large dipole magnets were removed from the accelerator to be measured and calibrated. The CED is being used to capture and disseminate key information generated during that process, such as new field maps for the magnets, locations where shims have been installed. During the same time period, software developers and operators updated many high-level applications (such as the Interactive Elegant Explorer shown in figure 4) and control system screens to utilize the CED for their configuration when operations resume.

The focus of the CED implementation will now shift to refining the tools and processes (automated where possible) that will keep the information up-to-date and accurate. In the case of the IOC information, for example, this will include building the DHCP server configuration files directly from the CED information as well as a program that crawls and inspects the IOCs periodically to verify their configuration against the database.

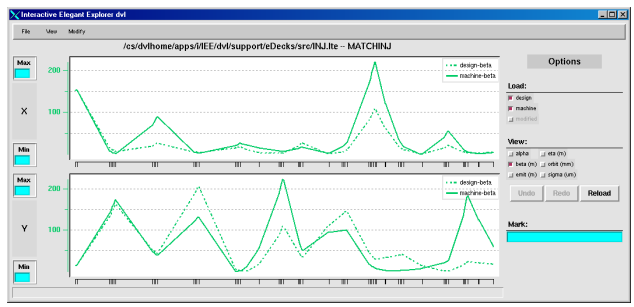


Figure 4: The Interactive Elegant Explorer now uses the CED to create a focusing lattice from a CEBAF configuration for the purpose of modeling accelerator beam transport.

REFERENCES

[1] Nadkarni, MD, Prakash M.; Marenco, MD, Luis; Chen, MD, Roland; Skoufos, PhD, Emmanouil; Shepherd, MD, DPhil, Gordon; Miller, MD, PhD, Perry (1999), "Organization of Heterogeneous Scientific Data Using the EAV/CR Representation", Journal of the American Medical Informatics Association 6 (6): 478–493, PMID 10579606

[2] "Oracle workspace Manager Overview." http://www.oracle.com/technetwork/database/twp-appdev-workspace-manager-11g-128289.pdf

[3] Beazley, David M (1996), "SWIG : An Easy to Use Tool For Integrating Scripting Languages with C and C++" Proceedings of the Fourth Annual Tcl/Tk workshop. Monterey, CA July 6-10.

[4] http://www.swig.org/

DIAMOND LIGHT SOURCE BOOSTER FAST ORBIT FEEDBACK SYSTEM

S. Gayadeen, S.R. Duncan, University of Oxford, Oxfordshire, UK,
C. Christou, M.T. Heron, J. Rowland, Diamond Light Source, Oxfordshire,UK

Abstract

The Fast Orbit Feedback system that has been installed on the Diamond Light Source Storage ring has been replicated on the Booster synchrotron in order to provide a test bed for the development of the Storage Ring controller design. To realise this the Booster is operated in DC mode. The electron beam is regulated in two planes using the Fast Orbit Feedback system, which takes the beam position from 22 beam position monitors for each plane, and calculates offsets to 44 corrector power supplies at a sample rate of 10 kHz. This paper describes the design and realization of the controller for the Booster Fast Orbit Feedback, presents results from the implementation and considers future development.

INTRODUCTION

The Diamond Storage Ring Fast Orbit Feedback (FOFB) system currently meets its requirements in terms of beam stability at Diamond. However new requirements for improved beam stability, from users or as a consequence of operating with reduced electron beam height or a need to suppress new beam disturbances in the future, will require improvements to the FOFB performance. As part of the development of the Storage Ring controller optimisation, closed loop beam control has been applied on the Booster synchrotron by running the Booster as an electron storage ring at 100 MeV. The Booster has the same hardware as the Storage Ring for beam position detection and control of the corrector magnets, so the same FOFB control system can be used.

CONTROLLER DESIGN

Let $\mathbf{P}(z^{-1})$ denote the discrete-time transfer function between $\mathbf{u}[k] \in \mathbb{R}^N$, the inputs to the $N = 22$ actuators applied at time $\{t = kT\}$, where T is the sample time, and $\mathbf{y}[k] \in \mathbb{R}^M$, the signals measured at the $M = 22$ sensors at each sample time, such that

$$\mathbf{y}[k] = \mathbf{P}(z^{-1})\mathbf{u}[k] \tag{1}$$

where $\mathbf{P}(z^{-1})$ takes the form

$$\mathbf{P}(z^{-1}) = p(z^{-1})\mathbf{B} \tag{2}$$

with $\mathbf{B} \in \mathbb{R}^{M\times N}$ being the steady state (dc) response of the actuators and $p(z^{-1})$ the scalar dynamics, which are taken to be the same for each actuator. The dynamics are modeled as a first order response plus delay that is operated in "sample and hold" mode, then

$$p(z^{-1}) = z^{-d}\frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} \tag{3}$$

where d is the smallest integer satisfying $dT > \tau_d$ and

$$\begin{aligned} a_1 &= e^{-aT} \\ b_0 &= 1 - e^{a(T-\tau')} \\ b_1 &= e^{a(T-\tau')} - e^{-aT} \end{aligned} \tag{4}$$

when τ_d is the delay in the system, a is the bandwidth of the actuator response (in rad.s^{-1}) and

$$\tau' = \tau_d - (d-1)T. \tag{5}$$

Table 1: Values of Parameters Used in Design

Parameter	Value
a	$2\pi\times1000$ rad.s^{-1}
τ_d	700 μs
T	100 μs

The parameters associated with this application are listed in Table so that the transfer function for the dynamic response becomes

$$p(z^{-1}) = z^{-7}\frac{0.47z^{-1}}{1 - 0.53z^{-1}}. \tag{6}$$

For $M \leq N$, the singular value decomposition of $\mathbf{B}$ takes the form

$$\mathbf{B} = \mathbf{U}\,[\Sigma\ \ 0]\,\mathbf{V}^T \tag{7}$$

where $\mathbf{U} \in \mathbb{R}^{M\times M}$ and $\mathbf{V} \in \mathbb{R}^{N\times N}$ are respectively, the left and right singular vectors and $\Sigma \in \mathbb{R}^{M\times M}$ is a diagonal matrix containing the singular values, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_M$. By partitioning $\mathbf{V}$ as

$$\mathbf{V} = [\mathbf{V}_1\ \mathbf{V}_2] \tag{8}$$

where $\mathbf{V}_1 \in \mathbb{R}^{N\times M}$ and $\mathbf{V}_2 \in \mathbb{R}^{N\times(N-M)}$, so that $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}_1^T$, then Eq. (1) can be written as

$$\mathbf{U}^T\mathbf{y}[k] = p(z^{-1})\Sigma{\mathbf{V}_1}^T\mathbf{u}[k]. \tag{9}$$

Defining $\bar{\mathbf{y}}[k] = \mathbf{U}^T\mathbf{y}[k]$ and $\bar{\mathbf{u}}[k] = \mathbf{V}_1^T\mathbf{u}[k]$, projects the response into "modal space", so that

$$\bar{y}_n[k] = p(z^{-1})\bar{u}_n[k] \tag{10}$$

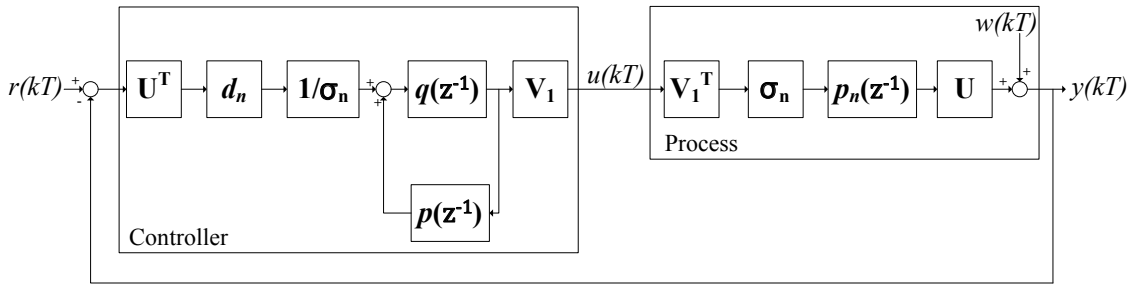


Figure 1: Structure of IMC controller.

where $\bar{y}_n[k]$ and $\bar{u}_n[k]$ are the n^{th} elements of $\bar{\mathbf{y}}[k]$ and $\bar{\mathbf{u}}[k]$. The controller structure is shown in Fig. 1 where an Internal Model Controller (IMC) [1] is applied to each mode where $q(z^{-1})$, the pseudo plant inverse dynamics is given by

$$q(z^{-1}) = \frac{(1-\lambda)}{b_o + b_1}\frac{1 - a_1 z^{-1}}{1 - \lambda z^{-1}} \tag{11}$$

and $\lambda = e^{-\zeta T}$ where ζ is the closed loop bandwidth (in rad.s^{-1}) of the controller. In [2, 3] it is proposed that the controller should take the form

$$\mathbf{C}(z^{-1}) = c(z^{-1})\mathbf{C} \tag{12}$$

where $\mathbf{C} \in \mathbb{R}^{N\times M}$ is

$$\mathbf{C} = \mathbf{V_1}\mathbf{D}\Sigma^{-1}\mathbf{U}^T \tag{13}$$

where

$$\begin{aligned}\mathbf{D} &= \text{diag}\{d_n\}\\ \Sigma &= \text{diag}\{\sigma_n\}\end{aligned} \tag{14}$$

with

$$d_n = \frac{\sigma_n^2}{\sigma_n^2 + \mu} \tag{15}$$

From Fig. 1 the controller dynamics for each mode are

$$\begin{aligned}c(z^{-1}) &= \frac{q(z^{-1})}{1 - p(z^{-1})q(z^{-1})}\\ &= \frac{(1-\lambda)}{b_0 + b_1}\frac{1 - a_1 z^{-1}}{1 - \lambda z^{-1} - z^{-d}(1-\lambda)\beta(z^{-1})}\end{aligned} \tag{16}$$

with

$$\beta(z^{-1}) = \beta_0 + \beta_1 z^{-1} = \frac{b_0 + b_1 z^{-1}}{(b_0 + b_1)}. \tag{17}$$

The same controller dynamics are applied to each spatial mode, with the dynamics being detuned by the controller gains d_n. Because $\beta_0 + \beta_1 = 1$, the controller takes the form of a Dahlin controller [4] and as a result, includes integral action.

CONTROLLER IMPLEMENTATION

Electron BPMs are used to provide information about the electron beam position at a sampling rate of 10 kHz. The Booster has 22 cells arranged as 4 sectors each of which has a computation node which receives all sensor positions but only calculates the local corrector magnet error using a block of the inverse response matrix. To achieve the required 10 kHz update rate, a custom communication controller implemented in VHDL is used to transmit the horizontal and vertical data from the 22 BPMs to each of the 4 computation nodes. Each computation node receives data from all BPMs and uses a dedicated VME processor card to calculate the vector product of the BPM values and the sub pseudo-inverse response matrix. The controller dynamics are then implemented as an eighth order IIR filter on these values. The result then corresponds to the new values for the local corrector magnets for that sector [5].

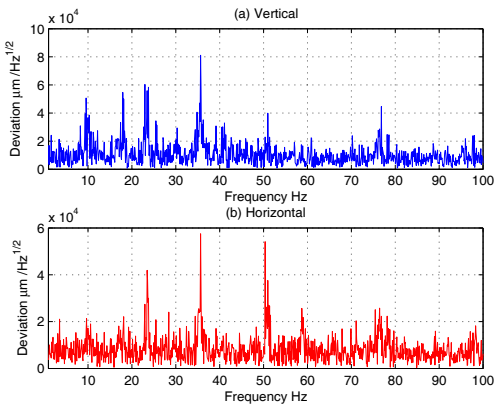


Figure 2: Magnitude of frequency response of 10s of variation observed at the 1st vertical BPM plotted against frequency in Hz.

CONTROLLER PERFORMANCE

The frequency content of the disturbance at one BPM on the Booster is shown in Fig. 2, which plots the discrete Fourier transform of the signal against frequency. It can be seen that there is a significant component in the range 10 Hz to 50 Hz with a major peak at 35 Hz. Fig 3(a) shows a colourmap of the average power (in dB) at each frequency in mode space and it can be seen that the bulk of the power is concentrated in the lower order modes (i.e. for n <10).

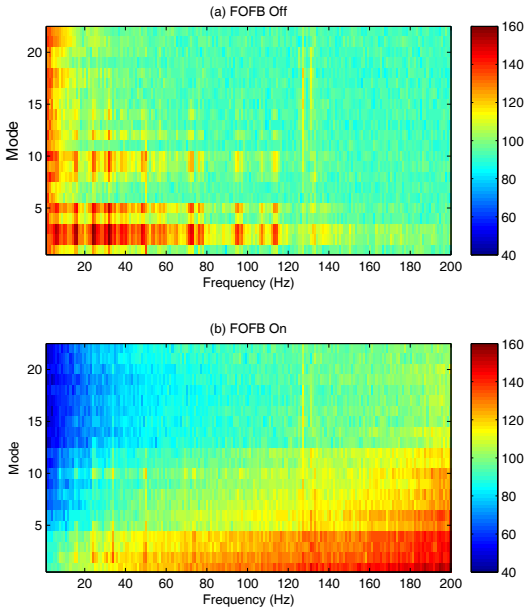


Figure 3: Colourmap of average power (in dB) in mode space against frequency (in Hz) in the vertical plane with (a) FOFB off and (b) FOFB on.

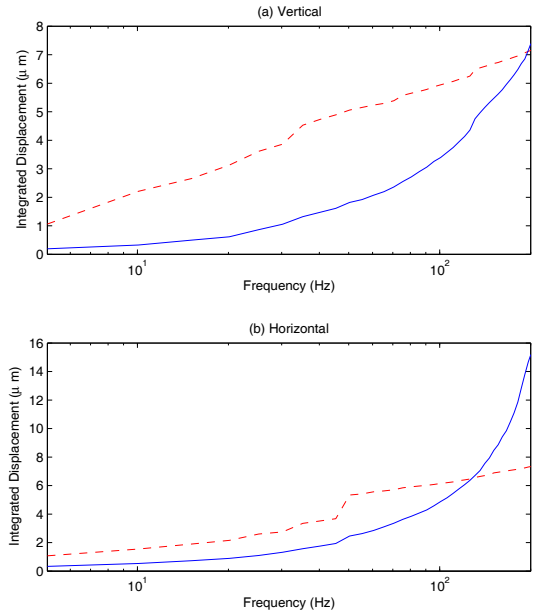


Figure 4: Integrated beam motion for controlled (solid blue line) and uncontrolled (dashed red line) beam up to 200 Hz.

There is also some variation at low frequencies (<1 Hz) in all modes.

The aim of the control system is to reduce the effect of the disturbances on the beam, particularly for the low order modes in the frequency range 10 Hz to 50 Hz, while at the same time attenuating low frequency disturbances in all modes. Figure 3(b) shows the power spectrum with FOFB on and by comparison with Fig. 3(a), it can be seen that the attenuation is focused on the low frequencies for all modes (i.e. the dark blue regions). The controlled and uncontrolled integrated beam motion for both planes, shown in Fig. 4, illustrates that the FOFB suppresses beam motion in the low frequencies where the disturbances are concentrated. However the FOFB system does not suppress disturbances as well at frequencies above to 200 Hz in both planes.

TWO DIMENSIONAL LOOP SHAPING

When a disturbance $\mathbf{w}[k]$ is included in the output $\mathbf{y}[k]$, then by defining $\overline{\mathbf{w}}[k] = \mathbf{U}_1^T \mathbf{w}[k]$, for each mode, $s_n(z^{-1})$, the transfer function from $\overline{\mathbf{w}}[k]$ to the output, $\overline{\mathbf{y}}[k]$ i.e. the sensitivity function is

$$s_n(z^{-1}) = \frac{1 - \lambda z^{-1} - z^{-d}(1-\lambda)(\beta_0 + \beta_1 z^{-1})}{1 - \lambda z^{-1} - z^{-d}(1-d_n)(1-\lambda)(\beta_0 + \beta_1 z^{-1})}$$

which can be used to analyse the behaviour of the closed loop system in Fig. 1. Figure 5 shows the Booster sensitivity both spatial and temporal dimensions $S(n, e^{-j2\pi\omega})$. The plane in Fig. 5 shows where $|S(n, e^{-j2\pi\omega})| = 0.7071$ i.e. the crossover frequency in both dimensions.

The bandwidth in the two-dimensional frequency domain can be similarly defined as traditionally done in the temporal frequency domain so that the closed loop bandwidth $\mathbf{B}_w(n,\omega)$ is defined as the contour in (n,ω) such that $|S(n, e^{j2\pi\omega})| = \frac{1}{\sqrt{2}}$ for all (n,ω) enclosed by $\mathbf{B}_w(n,\omega)$ [6]. The two-dimensional bandwidth frequency bandwidth contour is shown in Fig. 6 which also shows the effect of changing the regularisation parameter, μ and the closed-loop bandwidth, ζ. Figure 6(a) illustrates that increasing μ, focuses control effort on the lower order modes and decreasing μ, extends control to the higher order modes. It can also be seen that although μ has a significant impact on the spatial bandwidth, in this case it has no effect on the temporal component of the two-dimensional bandwidth $\mathbf{B}_w(n,\omega)$. Figure 6(b) illustrates that decreasing ζ, reduces the bandwidth to attenuate disturbances and has a negligible effect on the spatial component of the bandwidth $\mathbf{B}_w(n,\omega)$. From Fig. 6(b) it can be said that decreasing ζ has the effect of depressing the peak of the sensitivity function while reducing its temporal bandwidth.

The expected value of total power in the output is

$$E\|\mathbf{y}[k]\|_2^2 = \frac{1}{K}\sum_{k=0}^{K-1}\|\mathbf{y}[k]\|_2^2 \qquad (18)$$

In [2], it is shown that the expected total power can be written as

$$E\|\bar{\mathbf{y}}[k]\|_2^2 = \frac{1}{K^2}\sum_{p=0}^{K-1}\sum_{n=1}^{N}|S_n[p]|^2|\bar{W}_n[p]|^2 \qquad (19)$$

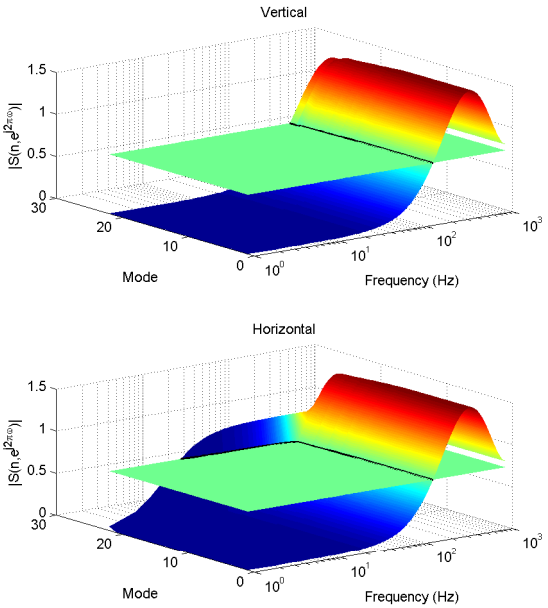


Figure 5: Magnitude of sensitivity ($|S(n, e^{i2\pi\omega})|$)for each mode against frequency (in Hz). The plane for $|S(n, e^{j2\pi\omega})| = 0.7071$ and the two-dimensional bandwidth contour (black line) are shown.

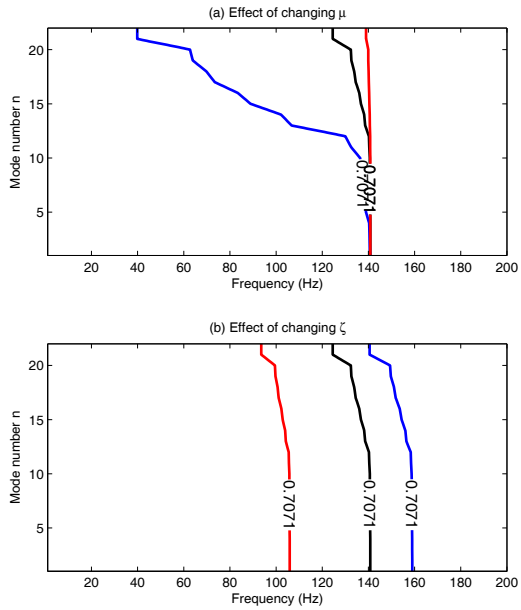


Figure 6: Two-dimensional bandwidth for (a) μ =0.1 (red), μ =1 (black) and μ =20 (blue) (b) ζ =303 Hz (red), ζ =227 Hz (black) and ζ =129 Hz (blue).

where $S_n[p]$ is the frequency response of $s_n(z^{-1})$. Eq. (19) implies that given $\bar{W}_n[p]$, the frequency spectrum of each mode of the underlying disturbance, $S_n[p]$ (and hence $s_n(z^{-1})$) can be designed on a "mode-by-mode" basis in order to achieve a specification on the expected total power observed across all sensors.

CONCLUSIONS & FUTURE WORK

FOFB control has been implemented on the Booster synchrotron at Diamond. The controller has been tuned conservatively and initial results show that the controller attenuates disturbances below 100 Hz on all modes.

The loop shaping technique developed by [6] shows that by extending traditional loop shaping techniques for dynamical systems into the two dimensional frequency domain, a framework to analyse the properties of the sensitivity function in the two-dimensional frequency domain analysis can be constructed. The sensitivity function properties in both temporal and spatial modes are used to define a two-dimensional bandwidth.

It was illustrated that tuning the temporal and spatial components of the sensitivity can be decoupled by using the dynamic closed loop bandwidth, ζ for shaping the temporal frequency response and the regularisation parameter, μ for the spatial frequency response. By considering the residual power in the output, it was demonstrated that given the frequency spectrum of each mode of the underlying disturbance, the sensitivity function can be designed on a mode-by-mode basis in order to achieve a specification on the expected average power across all sensors.

REFERENCES

[1] M. Morari and E. Zafiriou. *Robust Process Control*. Prentice Hall, 1989.

[2] S.R. Duncan. The design of a fast orbit beam stabilisation system for the diamond synchrotron. Technical Report 2296/07, University of Oxford, 2007.

[3] A. Napier, S. Gayadeen, and S.R. Duncan. Fast orbit beam stabilisation for a synchrotron. In *IEEE Multi-Conference on Systems and Control*, Denver, CO, 2011.

[4] D. Seborg, T. Edgar, D. Mellichamp, and F. Doyle III. *Process Dynamics and Control*. John Wiley and Sons, 2011.

[5] J. Rowland, M.G. Abbott, R. Bartolini, J.A. Dobbing, M.T. Heron, I. Martin, G. Rehm, I. Uzun, and S.R. Duncan. Status of the diamond fast orbit feedback system. In *Proc. Int. Conf. on Accelerator and LArge Experimental Physics Control Systems*, Knoxville, TN, 2007.

[6] G.E. Stewart. Two-dimensional loop shaping. *Automatica*, 39:779–792, 2003.

HIGH RESOLUTION ION BEAM PROFILE MEASUREMENT SYSTEM

José Gabriel Lopes[1,2,3,#], Jorge Rocha[3], Luís Manuel Redondo[1,2] and F. Corrêa Alegria[4]

[1] Instituto Superior de Engenharia de Lisboa, Portugal
[2] Centro de Física Nuclear da Universidade de Lisboa, Portugal
[3] Instituto Tecnológico e Nuclear, Sacavém, Portugal
[4] Instituto Superior Técnico/Universidade Técnica de Lisboa
and Instituto de Telecomunicações, Lisboa, Portugal

Abstract

A high resolution system designed for easy and inexpensive profile of an incident beam on a charged target, in the ion implanter installed at the Ion Beam Laboratory of the Technological Nuclear Institute (ITN) in Lisbon, Portugal, is described. This system intended to be used offline and not during the implantation, is made of a circular aluminium disc with a curved slit which extends approximately from the centre of the disc to its periphery. The disc is attached to the ion implanter target, which is capable of rotating on its axis. A cooper wire, positioned behind the slit, intersects the beam and the electric current generated, proportional to the beam intensity, is measured. As the ion implanter is capable of scanning the beam over the target, the combination of vertical beam scanning with aluminium disc rotation allows the beam profile to be measured continuously in two dimensions.

This paper describes the system developed including the computer controlled positioning of the beam over the moving curved slit, the data acquisition, the beam profile representation, and shows experimental results obtained with an Argon beam.

INTRODUTION

Traditional beam profile monitors use a matrix of detectors like Faraday Cups. The objective of this work is to develop a high resolution system for measuring the profile of an ion beam in an ion implanter. Two systems are in general use to monitor the beam profile, wire scanners and scintillation screens [1-3]. Both systems have proven their functionality and are chosen depending on the needs of the particular application. Their basic principles are well known and can be found in [4]. A third technique also used is residual gas monitors with segmented residual gas ion detecting plates, that is based on a similar principle as wire scanners regarding the reconstruction of the profile from the measured data.

Another technique is scintillation screens, which operates very fast and at very high resolution in position. They offer a convenient way to image the beam profile. Being their functionality based on (iono) luminescence [5], the most sensitive screens are made crystals like SiO_2. Using this option, the changes on the ion beam parameters influences the relative intensity of the intrinsic and activated luminescence [5]. Since the light yield also decreases with operational time, this method does not provide absolute values for the beam current. Also the lifetime of these kinds of detectors is strongly limited at higher beam powers. In such cases tantalum foils are used to illustrate the beam profile.

Absorbing the beam power a profile-dependent heat-induced glow is visible. To minimize the structure-smoothening and image-broadening heat flux inside the foil, the foils are chosen to be very thin. Nevertheless, their resolution is distinctly lower than that of crystal based scintillators. More examples for this detection system can be found in [1-2].

Wire scanners offer a quasi-non-destructive way to measure the profile in absolute current values. Their resolution and measuring speed generally behave antiproportional and depend on the number of wires used for detection and the analyzing electronics. A reliable determination of the beam profile can only be achieved if the ion distribution in the beam is Gaussian-like or at least single-peaked with only moderate asymmetry.

Problems occur if one unknowingly samples a hollow beam or a beam with more than one maximum in the current distribution. Additional details about this detection system can be found in [3, 6].

The presented system scans the beam on the target, using an off line beam profile scheme.

SYSTEM DESCRIPTION

As part of the deceleration system developed for the ion implanter, a beam profile monitor system was build to measure the shape of the beam over the target and set the optimal parameters for the deceleration lens taking into account the desired target bias.

The system developed, shown in Figure 1a, is composed by an insulated metallic disc of 25 cm diameter, which will be used as target, and a copper wire.

The target disc is mechanically attached to the implanter target, so it has the same rotating movement. On this target disc, a 3 mm wide and 10 cm long slit gap with a curved shape was made. A 3 mm wide and 10 cm long copper wire is placed behind the gap. This wire is fixed to the chamber structure and is connected to a microammeter, as shown in Figure 1b.

jgabriel@deea.isel.ipl.pt

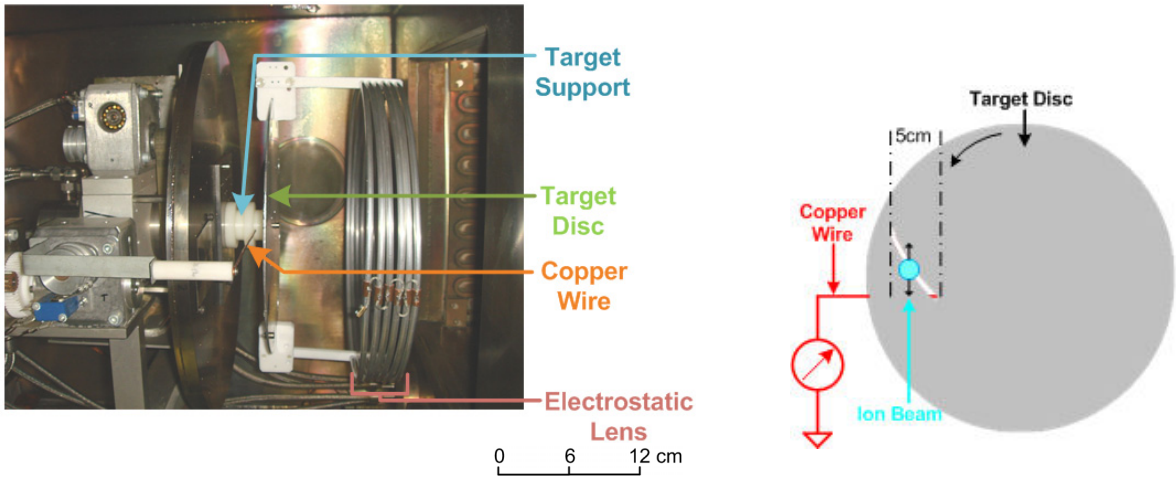


Figure 1: a) Mechanical system implemented; b) Schematic system implemented.

According to Figure 2, when the beam hits the target disc, a small part of it hits the wire and creates an electron current which is measured.

Due to the slit gap shape and the fact that the wire is fixed, when the target disc rotates, the beam strikes different parts of the wire in a way that it scans the beam in an area of 3x3 mm^2, corresponding to the slit gap x wire width, over a 5 cm distance, as shown in Figure 1b.

With the wire fixed on the horizontal, a focused beam can be scanned by rotating the target with the slit gap crossing the beam. If the beam current measured on the wire is collected one gets a beam profile section in *x*. By scanning the beam in y using the implanter beam scanning systems one gets x and y profile of the beam.

The reason to use the beam scan in y and not in x is that the drift of the beam due to the target bias changes with the distance to the edge of the target, but has no significant change if that distance is maintained.

In order to obtain an ion beam profile, two signals are required, one proportional to the beam width (x-axes) and another proportional to the beam current (y-axes).The beam current can be defined as

$$I_{Beam} = \frac{V_A}{47M\Omega} \quad (1)$$

Therefore, an optic-fiber based circuit was developed to send the beam current signal to the ground potential.

Whereas, on the resistance of 47 MΩ, the DC voltage proportional to the beam current is converted through a voltage-to-frequency (V/F), in a number of voltage pulses with a frequency proportional to its amplitude.

This signal is applied to an infrared LED connected to one side of an optic fiber. On the other side, placed at ground potential, a photo-transistor converts the light pulses to a number of voltage pulses with the original frequency. This signal is fed to a frequency-to-voltage (F/V) which is then digitized and goes to the personal computer (PC) as shown in Figure 2.

Figure 2 also shows the two high voltage (20 kV) power supplies used. One bias the lens (V_L) and the other bias the target (V_A). The resistances of 100 MΩ and 25 MΩ will allow a better stabilization of the voltage, avoiding any voltage fluctuations, while resistance of 47 MΩ is intended to measure the beam current.

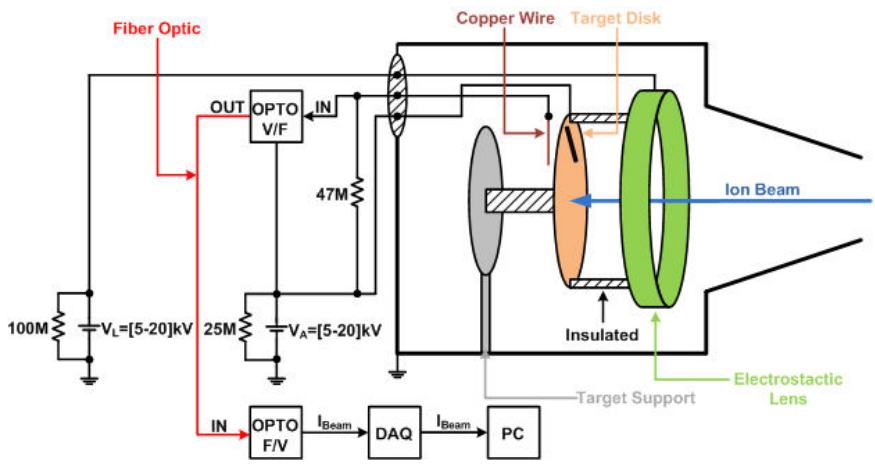


Figure 2: System developed.

A LabVIEW application was developed which controls the target position, gathers the digitized signals and determined the beam profile.

EXPERIMENTAL RESULTS

The system uses a personal computer equipped with a multifunction input/output board from National Instruments; model USB-6251. The program that was chosen for this work was National Instruments LabVIEW, a graphical programming language specially created for instrumentation and measurement, some custom made electronic interface modules and a LabVIEW application. Been integrated in a major project of the automation of the ion implanter, this is a part of the system that is considered as one of the most significant and that brings a great development and enhance to the ion implanter.

The ion beam profile measurement system developed makes integrant part of the optimization to the ion implantation, which also contemplates the deceleration of the ion beam to low energies. Checking the beam profile is important to guarantee that the implantations are successful even at low energies. In particular, as a result, it was possible to determine that the electrostatic deceleration developed for this accelerator is able to reduce the beam energy while maintaining the beam focused.

The experimental results presented in this work correspond to an ion implantation of Argon, with a beam current of 100 μA, initial energy of 15 keV.

The area of scan (x and y) is 5x7 cm^2, and the scan x corresponds to the rotational steps of the target. We made four experimental test: (1) with the target at 10 kV and the electrostatic lens with 0 kV (Figure 3a); (2) with the target with 10 kV and the electrostatic lens with 12 kV (Figure 3b), (3) with the target at 14 kV and the electrostatic lens with 0 kV (Figure 3c); (4) with the target at 14 kV and the electrostatic lens with 16 kV (Figure 3d). For the first and second test, the electrostatic lens is not charged, which means that ion beam is more scattered and less homogeneous. In the third and fourth test it is possible to see that the ion beam is more focused on the target. These two examples (Figures 3c and 3d), shows that the use of an electrostatic lens in this system gives an ion beam more focus and uniform.

In Figure 3d is possible to verify with more precision the effect of the electrostatic lens in the low energies (1 keV). In this case, the introduction of an electrostatic lens allows to obtain an ion beam more focused for low energies. When the electrostatic lens is not used, the ion beam is too disperse, which means that is not possible to proceed with the implantation for low energies (Figure 3c). Hence, the beam focus is directly related to the deceleration process, in order to obtain the desired final energy. This means that the bigger the deceleration level the more important is the beam focus.

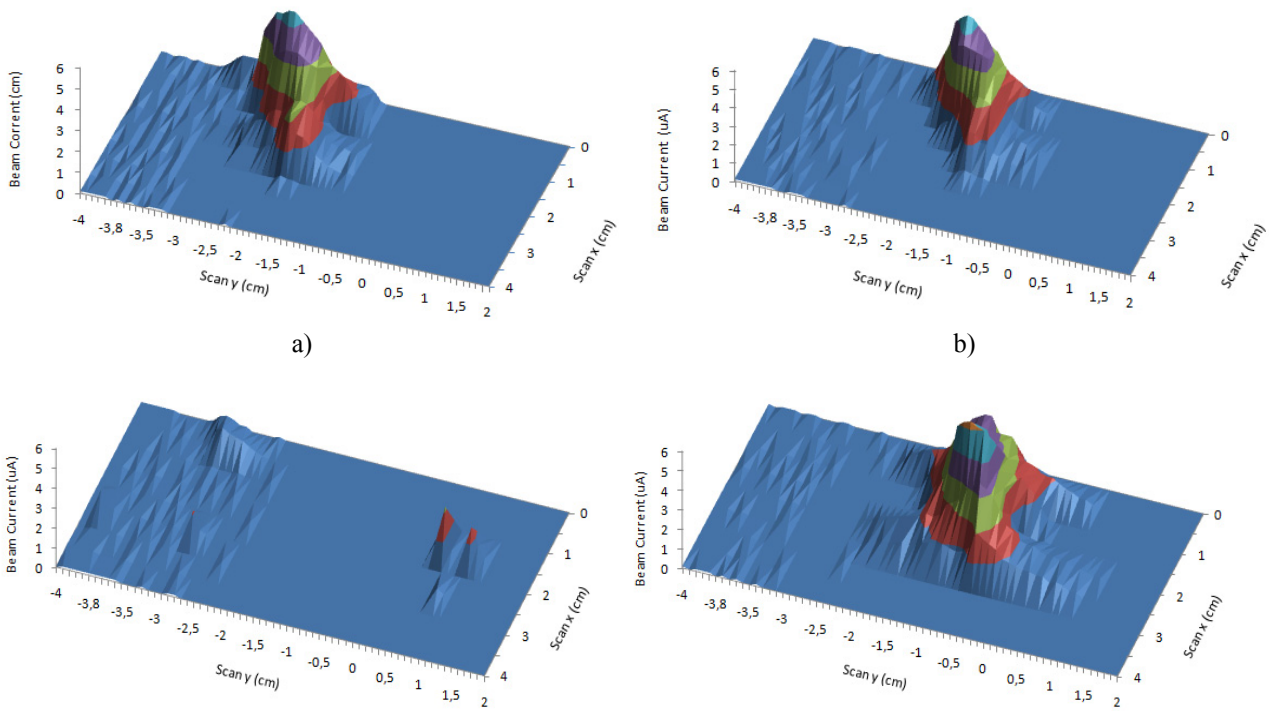


Figure 3: Beam profile for: a) E=5 keV (Target=10 kV, Lens=0 kV); b) E=5 keV (Target=10 kV, Lens=12 kV); c) E=1 keV (Target=14 kV, Lens=0 kV); d) E=1 keV (Target=14kV, Lens=16 kV).

CONCLUSION

In this paper a high resolution system designed for measuring the ion beam profile in the ion implanter installed at the Ion Beam Laboratory of the Technological Nuclear Institute (ITN) was described. The low cost and simplicity is the Figure of merit of this system.

With the system developed it is possible to measure the beam profile and to compare the differences between ion implantation based on energy and beam focus.

Detectors were used in order to be able to visualize the beam profile, in the systems there was presented. This allows obtaining tridimensional pictures of the ion beam with ion currents in the range of μA, and energies from 10 keV up to a few hundreds keV. Comparing this system with others two major differences come up, the current range and the type of implantation.

In addition is possible to obtain data for deceleration, in the future, the system will allow to test new configurations for beam focus at lower energy.

ACKNOWLEDGMENTS

JL thanks Fundação Calouste Gulbenkian for support. This work was supported by the Technological and Nuclear Institute, ITN, Sacavém, Portugal, and the Nuclear and Physics Center from the Lisbon University, CFNUL, Lisbon, Portugal.

REFERENCES

[1] D. S. Todd, D. Leitner, and M. Strohmeier, Low Energy Beam Diagnostics at the Venus ECR Ion Source, Beam Instrumentation Workshop 2008, Lake Tahoe, CA, 2008.

[2] P. Spädtke, R. Lang, J. Mäder, J. Rossbach, K. Tinschert, and J.Stetson, Ion Beam Extracted from a 14 GHz ECRIS of Caprice Type 17th Workshop on ECR Ion Sources and their Applications, ECRIS 06, Chinese Academy of Sciences, HEP&NP, 2007.

[3] T. Iida, Y. Maekawa, R. Taniguchi, M. Byakuno, and K. Sumita, Rev. Sci. Instrum. 52, 1328; 1981.

[4] Handbook of Ion Sources, edited by B. Wolf; CRC, NY, 1995.

[5] M. Suchanska, A. I. Bazhin, and E. I. Konopelko, Phys. Status Solidi B182, 231; 1994.

[6] G. Stover, IEEE Trans. Nucl. Sci. 32, 1988; 1985.

NSLS-II BEAM DIAGNOSTICS CONTROL SYSTEM*

Yong Hu#, Leo Bob Dalesio, Om Singh, Huijuan Xu, Kiman Ha, BNL, NSLS-II, NY 11973, U.S.A.

Abstract

A correct measurement of NSLS-II beam parameters (beam position, beam size, circulating current, beam emittance, etc.) depends on the effective combinations of beam monitors, control and data acquisition system and high level physics applications. This paper will present EPICS-based control system for NSLS-II diagnostics and give detailed descriptions of diagnostics controls interfaces including classifications of diagnostics, proposed electronics and EPICS IOC platforms, and interfaces to other subsystems. Device counts in diagnostics subsystems will also be briefly described.

INTRODUCTION

The NSLS-II beam diagnostics and control system is designed to monitor the electron beam of NSLS-II accelerator complex [1]. The beam quality is measured by a variety of parameters such as bunch charge, bunch structure (filling pattern), beam position/orbit, beam size/profile, energy & energy spread, circulating beam current, tunes, beam emittance, bunch length and beam losses. Figure 1 briefly shows the beam parameters to be measured from Linac to Storage Ring.

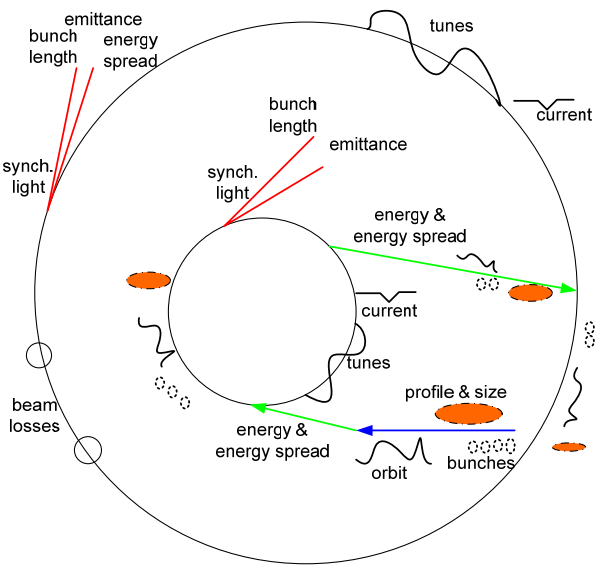


Figure 1: Beam Parameters to be measured at NSLS-II.

A correct measurement of beam parameters depends on the effective combinations of a variety of beam monitors, control and data acquisitions (DAQ) and high level physics applications. Thus, an effective collaboration between Diagnostics Group, Controls Group and Physics Group is essential. The NSLS-II Diagnostics & Instrumentation Group is in charge of the definition of beam diagnostics specifications and the design of beam monitor systems, including the layout of these monitors around the machine. The Physics Group is responsible for submitting the physics requirements for diagnostics and control, system modelling, algorithm, etc.

The scope of Controls Group on beam diagnostics system includes the following: requirement analysis and specifications for the beam diagnostics controls; Make vs. buy analysis and decision of electronics for various beam monitors; Development of EPICS [2] drivers for the diagnostic monitors and system tests on the software and hardware; Design of interfaces between diagnostics controls and other subsystems (timing, machine protection system, etc) and system integration of them.

DEVICE COUNTS IN DIAGNOSTICS

Table 1: Device Counts in NSLS-II Diagnostics

	Linac	Ltb	Booster	BtS	SR
WCM	5				
Screen/Flag	6	9	6	9	3
BPM	5	6	37	7	240
Bergoz FCT		2	1	2	
Bergoz ICT		2		2	
Energy Slit		1		1	
Faraday Cup	1	2		1	
Bergoz DCCT			1		1
Streak Camera					1
Visible Light Monitor			1		1
Pinhole system					2
Tune Monitor			1		1
Transverse Feedback System					1
Beam Loss Monitor					5
Beam Scrapers					5
Photon/x-ray BPM					2 per front-end

NSLS-II accelerators consist of one injector and one storage ring (SR). According to the functionality as well as geographical distribution, the injector is divided into 4 subsystems: Linac, Linac to Booster (LtB) transfer line (including 2 beam dumps), Booster, Booster to Storage ring (BtS) transfer line (including 1 beam dump). Table 1 gives a summary of the diagnostic monitors distributed over the whole machine. Although Linac and Booster are turnkey solutions provided by vendors, BNL will specify

* Work performed under auspices of the U.S. Department of Energy
# yhu@bnl.gov

the requirements for diagnostics and controls and vendors implement them. For better standardization and maintenance, the same type of diagnostics (e.g. CCD cameras) used in different accelerator subsystems will be provided by the same manufacture (e.g. GE1290 by Prosilica) so that they almost have the same control interfaces and requirements.

CONTROLS INTERFACES FOR DIAGNOSTICS

Diagnostics controls are actually more about data acquisition (DAQ) than device control. Diagnostics control subsystem will conform to NSLS-II control system standards. It will be EPICS-based and the preferable operating systems for IOCs are RTEMS (Real-Time Executive for Multiprocessor Systems) and Linux/Debian. For VME-based controls, the CPU board will be standardized as Motorola MVME3100.

Whenever possible, diagnostics controls pursue the utilization of commercial off-the-shelf hardware to reduce cost as well to achieve better reliability. Although NSLS-II Linac and Booter are turn-key solutions provided by vendors, it's better to standardize the diagnostics controls for the whole machine. The following section describes the classified method for controls standardization.

Classifications of Control Interfaces

From point view of controls, the beam monitors output signals/interfaces can be classified into the following several groups.

1) Analog output with high-bandwidth (>500MHz): WCM, FCT, etc. [3];
2) Analog output with low-bandwidth (<10KHz): DCCT, ICT&BCM;
3) Simultaneous 4-channle RF signals: BPM;
4) Gigabit-Ethernet camera interface: pinhole camera, flag/CCD, streak camera etc.
5) Stepper motor driven: linear stage in pinhole system, energy slit, beam scraper, etc.
6) Ethernet-based instrument: Windows XP-based network/spectrum analyzer for tune monitor and beam stability monitor;

There are other miscellaneous I/Os for diagnostics: binary input/output including TTL I/O for DCCT range settings, pneumatic actuator with limit switch in flag, limit switch in stepper-based stage, 24 V binary outputs for XIA filter inserter in pinhole system, 12-bit DAC for illuminator control in pinhole and flag system, temperature sensors for diagnostics beamline mirror, etc.

Here are some considerations and principles about the selection of electronics for the above groups of beam monitors:

1) WCM and FCT are used to measure individual bunch charge. The time interval between adjacent bunches is 2 ns so that the digitizer for WCM & FCT should have at least 500MHz bandwidth and 1GS/s sampling rate. Since NSLS-II physics requirement for filling pattern accuracy is 1% of maximal bunch charge, 8-bit resolution digitizer should be fine;
2) The output of DCCT and BCM/ICT is nearly DC voltage so that the requirement of sampling rate for digitizer is not demanding. For Booster DCCT and transfer-lines ICT, the digitizer with 16-bit resolution and 20KS/s would be suitable. For Storage Ring DCCT digitizing, the resolution requirement is determined by beam lifetime calculation considered 20mA in 60 hours with 1% accuracy. This translates to 18-bit ADC resolution;
3) For 4-button-pickup RF BPM, in-house BPM receiver will be utilized;
4) For camera controls, NSLS-II diagnostics will be standardized as Gigabit-Ethernet interface (Prosilica Gig-E CCD camera) for high bandwidth, easy cabling, good anti-EMI, etc.;
5) For stepper-motor based diagnostics, the controls will be integrated into NSLS-II motion control subsystem;
6) For Windows-based network/spectrum analyzer, it's possible to make an EPICS IOC run inside the instrument.

Controls and Data Acquisitions for Diagnostics

Each type of beam monitor requires electronics (device controller) to process its output signal. The electronics for the above groups and associated EPICS IOC platform are listed in Table 2. Figure 2 shows the controls interfaces for these various beam monitors.

Table 2: Diagnostics Electronics and IOC Platform

Beam Monitor	Diagnostics Electronics	IOC platform
WCM & FCT & FC	Acqiris DC252 (2GHz bw, 10-bit, 4~8GS/s)	cPCI/Linux
DCCT & ICT	1)GE ICS-710-A (24-bit, 200KS/s, 8-ch) 2)Allen-Bradley PLC (DAC, Digital I/O)	cPCI/Linux
BPM	In-house BPM receiver [4]	PC/Linux
Prosilica GigE Camera	PC/Linux	PC/Linux
Stepper-motor-based	Delta Tau GeoBrick LV PC	PC/Linux
Instrument controls	Windows-based network/spectrum analyzer	PC/Linux
Misc: digital I/O, DAC, temperature sensor	Allen-Bradley PLC	PC/Linux

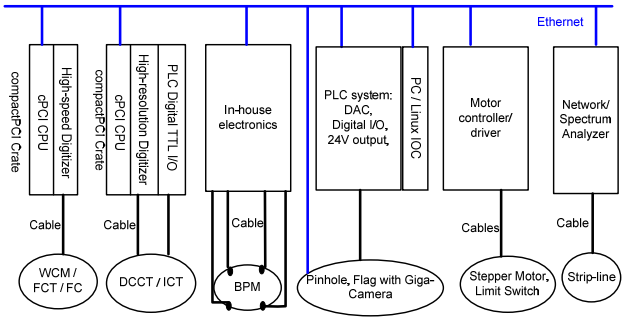


Figure 2: Diagnostics Controls Interfaces.

INTERFACES TO OTHER SUBSYSTEMS

Interfaces to Timing System

To capture the electron beam signal at the right time, the beam monitors should be sampled and synchronized to the passage of the beam. This function can be achieved by using Event Timing System to deliver delayed-trigger or clock signal to the diagnostics electronics.

The interfaces between diagnostics controls and timing system are shown in Figure 3. Some diagnostics controls, such as stepper motor, limit switch and DAC, don't need timing while other diagnostics electronics require trigger signals from Event Timing System [5] (EVR). Additionally, the reference for BPM receiver Machine Clock should be provided at the Booster/Storage Ring revolution frequency. For transverse bunch-by-bunch feedback system and streak camera system, trigger/clock signals are the only control signals.

For diagnostics timing requirements, 8ns resolution and 100ps jitter should be sufficient. Diagnostics controls require precisely time delayed trigger at injection rate (10Hz for Linac, 1Hz for Booster) to capture the beam at the right time. The delayed time for each diagnostics depends on the location of beam monitor, the cable length from monitor to its electronics and the distribution of event timing.

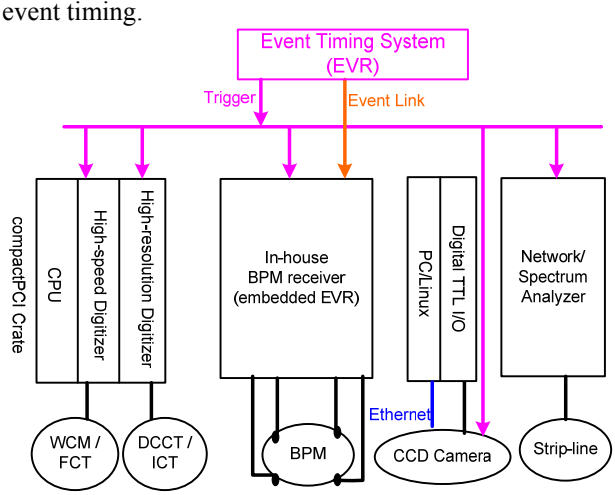


Figure 3: Diagnostics Interface to Timing System.

Interfaces to Machine Protection System

The diagnostics controls should send hardwired interlock signals, to machine protection system (MPS) if any specific parameter, such as beam positions, beam loss rate, beam current, is out of range.

The design of the interfaces between diagnostics controls and MPS are still in progress. Fig. 4 shows the draft diagram of the system.

The followings are some cases that diagnostics interlock protection should take action:

1) BPM position data are out of pre-defined position range;
2) For top off operation, the stored beam current must exceed 50mA;
3) Excessive beam loss is detected by LCM system;

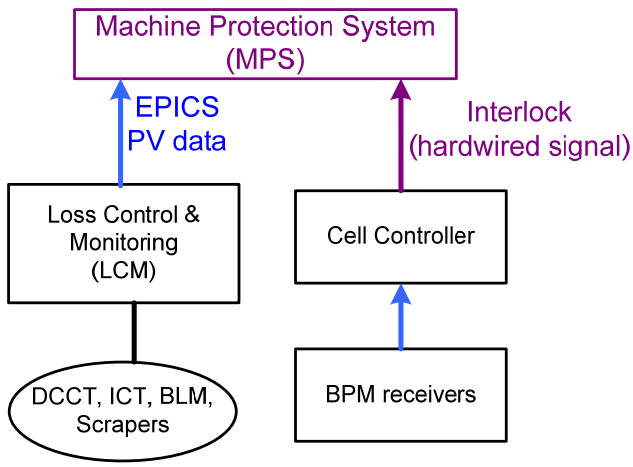


Figure 4: Interface to Machine Protection System.

CONCLUSIONS

NSLS-II beam diagnostics control system is completely EPICS-based. Utilization of commercial off-the-shelf hardware as well as in-house BPM electronics development is deployed to meet NSLS-II project requirements and schedule. All diagnostics controls hardware and software are well tested and ready for NSLS-II Injector installation and commissioning.

REFERENCES

[1] NSLS-II Preliminary Design Report", http://www.bnl.gov/nsls2/project/PDR.
[2] http://www.aps.anl.gov/epics/.
[3] Yong Hu, et al., "NSLS-II Filling Pattern Measurement", Proceedings of ICALEPCS 2011, Grenoble, France.
[4] Yong Hu, et al., "BPM Inputs to Physics Applications at NSLS-II", Proceedings of PAC 2011, New York, NY, US.
[5] http://www.mrf.fi/.

APPLICATION OF INTEGRAL-SEPARATED PID ALGORITHM IN ORBIT FEEDBACK *

K. Xuan#, X. Bao, C. Li, W. Li, G. Liu, J. Wang, L. Wang,
NSRL, USTC, Hefei, Anhui 230029, China.

Abstract

The algorithm in the feedback system has important influence on the performance of the beam orbit. PID（Proportion Integration Differentiation） algorithm is widely used in the beam orbit feedback system; however, the deficiency of PID algorithm is a big overshooting in strong perturbations. In order to overcome the deficiencies, the integral-separated PID algorithm is developed. When the closed orbit distortion is too large, it cancels integration action until the closed orbit distortion is lower than the separation threshold value. The implementation of integral-separated PID algorithm with MATLAB is described in this paper. The simulation results show that this algorithm can improve the control precision.

INTRODUCTION

The orbit feedback system can effectively eliminate the closed orbit distortion; a good feedback system can improve remarkably the beam orbit stability. PID algorithm is widely used in the orbit feedback system. In the standard PID feedback, the aim of introduction of the integral action is to eliminate steady-state error and improve control accuracy, but the integral action of PID algorithm will cause big overshooting in strong perturbations. In order to improve the orbit stability, an integral-separated PID algorithm is employed for the orbit feedback control. The basic idea of the integral-separated PID controller is that integral action will be cancelled when there is a big error. The integral action will work when the output value is close to the given value, which will eliminate steady-state error and improve control accuracy.

FEEDBACK THEORY

The beam orbit distortion caused by the change of storage ring state would destroy the stability of the beam orbit. There are many methods to correct the beam closed orbit distortion, but the basic idea of these methods is to eliminate the orbit distortion by using corrector magnet. In order to eliminate the beam orbit distortion, the corrector magnet intensity changes $d\Theta$ should be[1]:

$$d\Theta = R^{-1}dU \qquad (1)$$

where dU is the beam orbit distortion, R^{-1} is inverse response matrix, SVD（Singular Value Decomposition） method can be used to solve R^{-1}.

In order to correct the orbit distortion and improve orbit stability, the integral-separated PID control algorithm will be used. The control algorithm can be expressed as[2]:

$$u(n) = u(n-1) + K_p\left[e(n) + \alpha\frac{T}{T_I}\sum_{j=0}^{n}e(j) + T_D\frac{e(n)-e(n-1)}{T}\right] \qquad (2)$$

$$\alpha = \begin{cases} 1 & |e(n)| \le \varepsilon \quad \text{PID control} \\ 0 & |e(n)| > \varepsilon \quad \text{PD control} \end{cases}$$

where K_P is named as proportional coefficient, T_I ,T_D is named as integral and differential time constant separately, T is sampling interval, n is the sample number, $e(n-1)$ is the error signal for (n-1), $e(n)$ is the error signal for n, $u(n-1)$ is the output of the Integral-separated PID digital controller for (n-1), $u(n)$ is the output of the Integral-separated PID digital controller for n, ε is the artificial separation threshold.

The basic concept of the orbit feedback system is shown in the Figure 1[3][4]. The feedback controller is based on an Integral-separated PID algorithm.

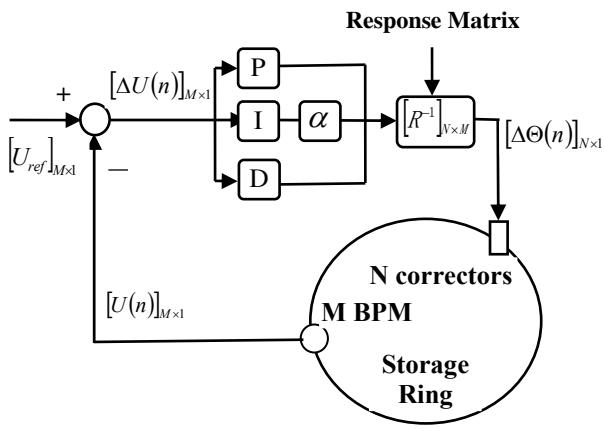


Figure 1: The schematic diagram of the integral-separated PID orbit feedback system.

Where U_{ref} is the reference orbit, $U(n)$ is the BPM value for n, $\Delta U(n)$ is the closed orbit distortion for n, $\Delta\Theta(n)$ is the corrector magnet intensity changes for n.

According to formula (1) and (2), we get the discrete integral-separated PID control formula of beam orbit correction:

* Work supported by National Natural Science Foundation of China (11005114)
#xuanke@ustc.edu.cn

$$\Theta(n) = R^{-1}U(n) = R^{-1}U(n-1) + R^{-1}K_P \times \left[\Delta U(n) + \alpha \frac{T}{T_I}\sum_{j=0}^{n}\Delta U(j) + T_D\frac{\Delta U(n) - \Delta U(n-1)}{T}\right] \quad (3)$$
$$= \Theta(n-1) + K_P\Delta\Theta(n) + \alpha K_I\sum_{j=0}^{n}\Delta\Theta(j) + K_D[\Delta\Theta(n) - \Delta\Theta(n-1)]$$

where $U(n-1)$ is the BPM value for (n-1), $\Delta U(n-1)$ is the closed orbit distortion for (n-1), $\Theta(n)$ is the corrector magnet intensity for n, $\Theta(n-1)$ is the corrector magnet intensity for (n-1), $\Delta\Theta(n-1)$ is the corrector magnet intensity changes for (n-1), K_I, K_D is named as integral and differential coefficient separately. $K_I = K_P T/T_I$, $K_D = K_P T_D/T$.

ORBIT FEEDBACK SIMULATION

The integral-separation PID feedback algorithm is studied in HLSII storage ring. The storage ring accelerator model is built by AT (Accelerator Toolbox), and the code of the integral-separation PID feedback algorithm is programmed by using MATLAB[5][6].

Concrete steps are introduced as in following:

(1) Obtain the response matrix by using AT function, and calculate the inverse response matrix R^{-1}.

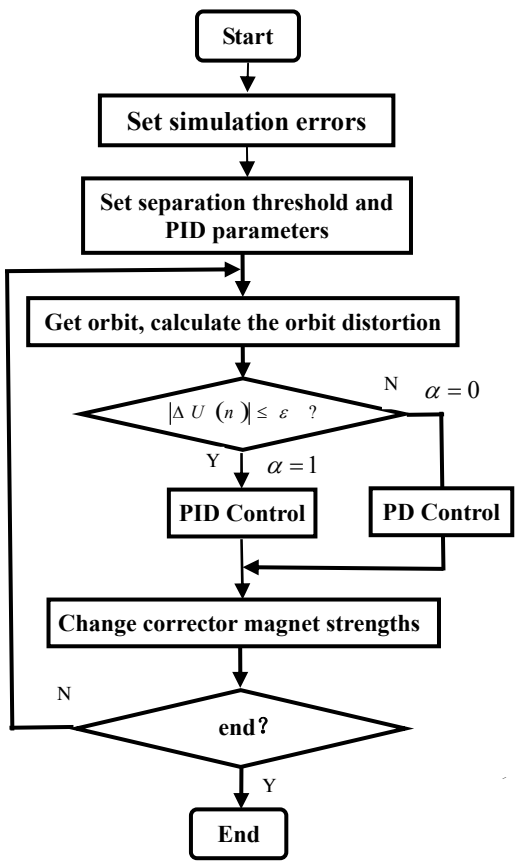


Figure 2: The flow chart of the integral-separation PID orbit feedback algorithm.

(2) Produce orbit distortion by simulating a variety of errors.

(3) Set separation threshold value and PID parameters according to the actual system.

(4) Calculate the closed orbit distortion $\Delta U(n)$, if $|\Delta U(n)| > \varepsilon$, Using PD feedback, if $|\Delta U(n)| < \varepsilon$, Using PID feedback.

Repeat 4th step until it reaches the target. Figure 2 shows the flow chart of the integral-separation PID orbit feedback algorithm.

After repeated simulation, the optimal PID control parameters and separation threshold value of the controller module can be determined:

$$K_P = 2, K_I = 0.4, K_D = 0.9, \varepsilon = 0.1mm \quad (4)$$

The effect of orbit feedback by using integral-separation PID algorithm are recorded and shown in figure 3 and figure 4.

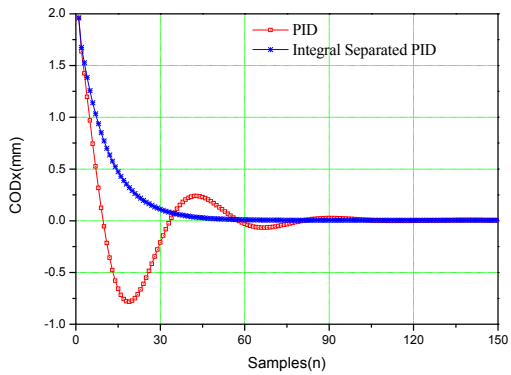


Figure 3: The horizontal orbit of PID control and integral-separation PID at the 16th BPM.

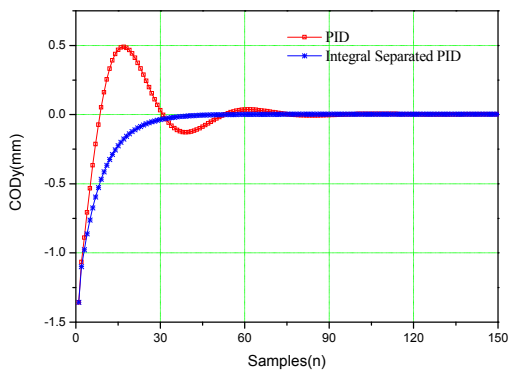


Figure 4: The vertical orbit of PID control and integral-separation PID at the 16th BPM.

Figure 3 and Figure 4 shows the beam orbit stability has been greatly improved after adding integral-separation PID feedback when the orbit distortion is large. After introducing integral action, the proportional coefficient K_P should be adjusted to ensure the stability of the feedback system.

CONCLUSION

The simulation results show the beam orbit feedback based on the integral-separation PID could be smoother and faster than normal PID control, and the control effect has been greatly improved. It is noteworthy that the separation threshold value should be determined according to the specific circumstances and requirements. If the separation threshold is too large, the integral-separation target can not be achieved; if the separation threshold is too small, it will not enter the integration area. If only the PD control, it will not to eliminate steady-state error and improve the beam orbit stability.

REFERENCES

[1] Li Jingyi, HLS Control System and Physical Research Based on Control System . Hefei: University of Science and Technology of China, p.70-90 (2002) .

[2] Araki M. PID control, Control Systems, Robotics, and Automation, Vol.II, p.1-23(1995).

[3] C.H. Kuo, Jenny Chen. "Orbit feedback system for the storage ring of SRRC", ICALEPCS'2001, Nov 2001, WEAP047, p. 373-375 (2001).

[4] Jeffrey A,Kirchman , Youngsoo Chung, "Experimental Results on the Design for the APS PID Global Orbit Control System" , ICALEPCS'95, Jan. 1995, W-PO-17,(1995).

[5] Terebilo A. "Accelerator toolbox for MATLAB".SLAC-PUB-8732, (2001).

[6] Liu Jingkun, "MATLAB simulation of advanced PID control", Beijing： Publishing house of electronics industry, p.27-32 (2011).

DESIGN OF A DIGITAL CONTROLLER FOR ALPI 80 MHZ RESONATORS

S. Barabin, Institute of Theoretical and Experimental Physics, Moscow, Russia
G. Bassato, INFN, Laboratori Nazionali di Legnaro, Legnaro (Padova), Italy

Abstract

We discuss the design of a resonator controller based mainly on digital technology. The signal frequency is 80 MHz but can be easily increased up to 350MHz; the controller can work in either "Generator Driven" (GDR) or "Self-Excited Loop" (SEL) modes. The signal processing unit is a commercial board (Bittware T2-Pci) with Xilinx Virtex II-Pro FPGA and 4 TigerSharc DSPs. The front-end board includes a set of A/D channels, which sample the RF signals coming from the cavity pickup and from the reference generator. We present results of some preliminary tests on 80 MHz quarter wave resonator installed in the ALPI Linac accelerator at INFN-LNL and discuss possible developments of this project.

INTRODUCTION

Low level RF control is nowadays implemented using digital techniques [1, 2]. The availability of high-density FPGAs commercial boards makes possible to implement in hardware computational functions that, in the past, would have required an array of high performance DSPs. The board we chose for our application contains both, but most of RF processing blocks (i.e. signal conversion and digital signal processing) are configured inside the FPGA, while DSPs and their memory are mainly used for data routing and several diagnostic calculations.

The general concepts and ideal characteristics of a digital controller are described in [2]. The main features currently realized in our RF controller can be summarized as follows:

- Capability of working in both SEL and GDR modes and switching between them;
- Capability of operating with both superconducting and room temperature cavities;
- Selection between I/Q and amplitude/phase control;
- Frequency sweep mode and frequency response plot generation;
- Provision for cavity frequency resonance control by means of mechanical tuner;
- Minimal use of analog components – no analog downconverters, vector modulators and phase shifters.

RF CONTROLLER DESCRIPTION

As shown in fig. 1 and fig. 2, the controller is based on three boards: the analog front-end for RF signal acquisition, the Bittware T2-PCI board [3] containing DSPs and FPGA, and an output board to generate the signal driving the power amplifier.

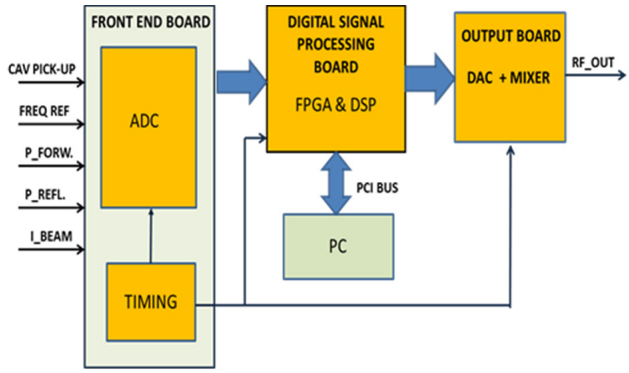


Figure 1: Resonator controller block diagram.

The analog front-end includes five A/D channels based on the AD9433 ADC. Two A/D sample RF signals coming from the cavity pickup and from the frequency reference generator. Additional ADC channels acquire information of amplifier's direct and reflected power and beam current, in order to extend the controller capability to accelerators with high current and beam loading effects. For each channel, an Epcos B39805 bandpass filter followed by a Mini-Circuits ERA-3 amplifier and an ADT1–1WT transformer provide the required filtering and impedance adaptation for the ADC input.

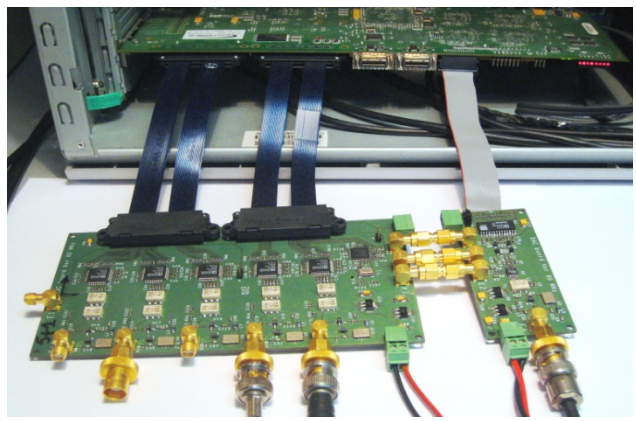


Figure 2: The controller boards.

The output signal driving the power amplifier is generated by a DAC (AD9752) at a frequency of 16MHz; after proper filtering this signal is then upconverted to 80 MHz by mixing with the 64 MHz local oscillator signal. Clock signals to 5 ADCs, DAC, mixer and FPGA are provided by the AD9516 programmable clock distributor, placed in the ADC board and featuring a channel-to-channel skew lower than 10ps. The frequency of sampling clock is 64MHz. The ADC and DAC boards are connected to the T2-PCI card through dedicated high

speed I/O channels, supporting parallel data transfer from all five 12-bit ADC and the 12-bit DAC at 64 MHz rate.

Signal Flow

Figure 3 shows the FPGA block diagram. I/Q demodulation is performed by sampling the input signals from ADCs at a frequency of 4/5 of reference signal to get a 90 deg. phase advance between two samples; the sampled I, Q components can be rotated by a rotation matrix in order to compensate the delay introduced by cables.

Digital downconversion and filtering are performed by cascaded integrator-comb (CIC) filters, to reduce signal noise. Then, signals are converted to polar form by the COordinate Rotation DIgital Computer (CORDIC) to allow phase and magnitude comparison with predefined values. The result of comparison (phase and amplitude errors) goes through a PID algorithm to generate correction signals; a further rotation on phase angle can be optionally added to compensate offsets.

Correction signals, in polar form, undergo a second CORDIC block to convert back to the complex amplitude. form. Resulting I and Q signals drive a digital quadrature modulator [4] that generates, at 16 MHz rate, patterns of 12 bit values representing samples of output signal in the time domain; these values are then sent to the output DAC.

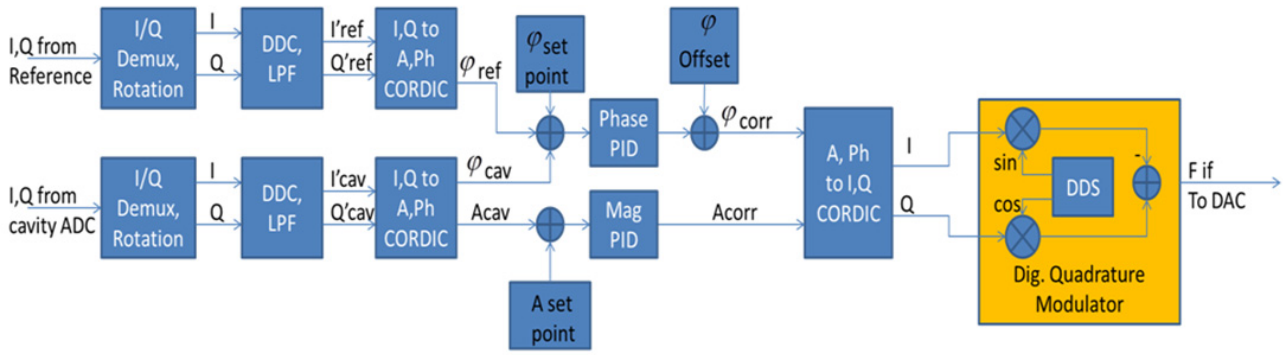


Figure 3: Signal flow inside the FPGA.

DESIGN TOOLS

The DSP code was generated by the C compiler included in Analog Devices VisualDSP++, while the FPGA project was compiled using Xilinx ISE Design Suite. Many FPGA functions are part of Xilinx System Generator module, which is a plug-in extension of Platform Studio SDK. This toolkit also supports the compilation of the PowerPC block, available in the VIRTEX-II Pro chip, we used to configure the clock generator. Matlab and Simulink (by Mathworks) were extensively used in conjunction with Xilinx System Generator module to compile and simulate several FPGA blocks.

USER INTERFACE

The program named "RF controller" implements the user interface to set and diagnose the RF field in cavity. The graphic application is written in C++ with Microsoft Visual Studio IDE. It uses Bittware's host interface library to communicate with Bittware's digital board through PCI.

User Interface Main Functions:

- Setting up desired output signal amplitude and phase;
- Switching on/off the amplitude and phase feedback;
- Select feedback type between I/Q and amplitude/phase;
- Tuning of feedback parameters like proportional and integral coefficients;
- Switching between SEL and GDR modes;
- Detection of the cavity's resonance frequency during start up (with SEL or amplitude-frequency plot);
- Tune trade-off between loop delay and noise of signal measurement, by tuning digital filters.

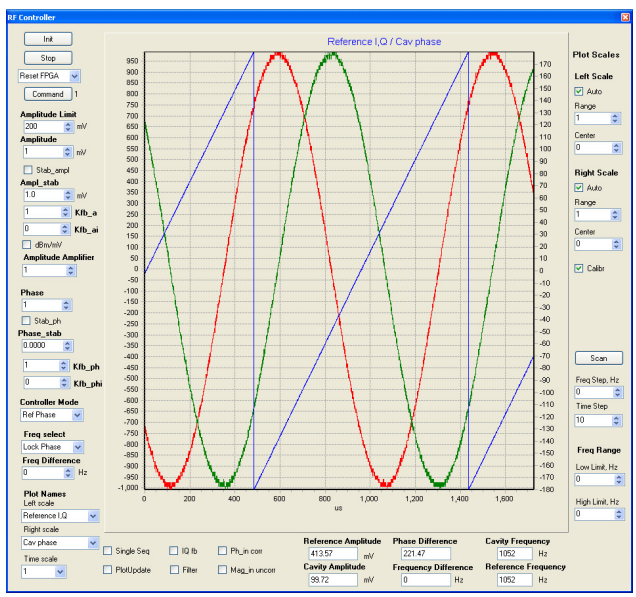


Figure 4: The "RF Controller" graphic interface.

Diagnostic Possibilities

A plot window can simultaneously present two graphs taken from different stages of signal flow. Graphs can be updated at a fixed rate, or triggered by a defined event (i.e. to view a step response). In addition, plot can show cavity's frequency response. Text windows below the plot show several parameters such as reference and cavity field amplitude, frequency and phase difference. These data are updated at approximately 8 Hz rate. Phase and frequency difference are sent by Ethernet to cavity tuner.

TEST RESULTS

Feedback Tests

Feedback capability was tested separately for amplitude and phase. Amplitude and phase PID gains have been tuned to find the optimal gain to minimize the transient response time without overshoot.

Currently, phase difference in the feedback mode can be changed with ~0.09° step, and phase difference textbox shows the same measurement value with 0.01° resolution. The signal amplitude in feedback mode can be changed with 0.1 mV (or ~0.05% from maximum value) step and measured with approximately the same resolution.

Frequency Response Tests

"RF Controller" program allows switching of the output signal frequency to the reference frequency, to the value of 5/4 of clock frequency and to any other desirable frequency (with 1 Hz step). Frequency response is achieved by automatic scan of output signal frequency.

Finding the Cavity Resonance Frequency

One possibility to find the resonance frequency on switch up is sweeping the output frequency and acquiring the cavity's response. It was shown that the reference frequency in test cavity was ~79.7 MHz, and halfwidth bandwidth was ~80 kHz.

Another possibility is using SEL mode. Before SEL mode switch-on difference between cavity and output frequency was about 300 kHz. After SEL mode activation, cavity was tuned to its resonance frequency, as visible on scope and on PC program. We also observed, after modifying cavity's resonance frequency by moving copper rings inside it, the controller was able to track frequency changes.

Feedback Tests with Cavity

It was shown that frequency locking in SEL mode is possible. Phase error in the SEL phase feedback mode was large, about 10 degrees, because of large bandwidth of the warm cavity. It is expected that the phase error for superconducting cavity can be much lower. After switching to the GDR mode (typical working mode), the phase error is reduced to 1%.

Figure 5a and 5b show the phase error under environmental disturbance conditions (mechanical vibrations) without feedback (5a) and with the cavity locked (5b) when the cavity works in generator-driven mode. The amplitude error with feedback is about 1 mV.

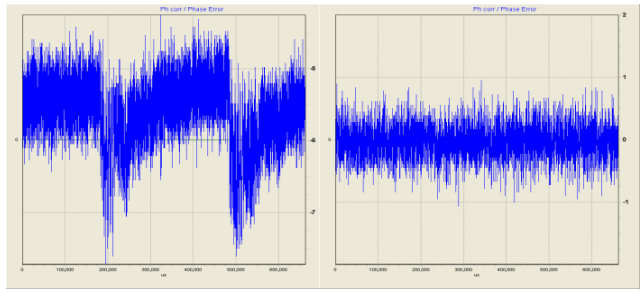


Figure 5a: cavity unlocked. Figure 5b: phase locked.

Digital Filters Addition

Amplitude and phase errors can be reduced by addition of digital filters. Using filters decreases the phase error from 1° down to ~0.25° by cost of increasing the loop delay: the loop delay, in fact, increases from ~5 to 60 µs, and the step response time grows up to 400 µs. For superconducting cavities with typical filling times up to 1 second, this loop delay is more than acceptable.

CONCLUSION

The prototype we realized demonstrates the feasibility of a digital solution fully based on FPGA. Although the preliminary results are promising, some work still need to be carried out. New features can be added by software modifications, such as:

- Feedforward beam loading;
- Power amplifier linearization;
- Any combination of amplitude, phase and I, Q control;
- Dynamic change of digital filter characteristics;
- Additional feedforward modes can be added – open loop operation at start-up, cavity conditioning/tuning, adaptive feedforward.

A significant part of hardware changes should be made by redesigning the analog section of controller, that includes: field calibration of cavity input, increase of input dynamic range by addition of variable gain amplifiers, increase of ADC resolution from 12 to 16 bit, increase of DAC resolution to 16 bit, other than its output level. Another possible change concerns with the digital board: a more modular design (i.e, changing the interface to the host computer from PCI to Ethernet) would reduce the cost, increase the flexibility and improve the performance.

REFERENCES

[1] C.Hovater et al. "A digital self excited loop for accelerating cavity field control", PAC'07.
[2] S.Simrock. "Universal Controller for digital RF Control", EPAC'06.
[3] www.bittware.com
[4] K. Gentile: "DDS simplifies polar modulation", EDN, August 2004, p69

FAST ORBIT CORRECTION FOR THE ESRF STORAGE RING

Jean Marc Koch, Francis Epaud, Eric Plouviez, Kees Scheidt, ESRF, Grenoble, France

Abstract

Up to now, at the ESRF, the correction of the orbit position has been performed with two independent systems: one dealing with the slow movements and one correcting the motion in a range of up to 200Hz but with a limited number of fast BPMs and steerers. The latter will be removed and one unique system will cover the frequency range from DC to 200Hz using all 224 BPMs and the 96 steerers. Indeed, thanks to the procurement of the Liberas Brilliance and the installation of new AC power supplies, it is now possible to access all the Beam positions at a frequency of 10 kHz and to drive a small current in the steerers in a 200Hz bandwidth. The first tests of the correction of the beam position have been performed and will be presented. The data processing will be presented as well with a particular emphasis on the development inside the FPGA.

INTRODUCTION

The ESRF storage ring is a high brilliance source with low emittance values (ε_x =4.10^{-9} m.rad and ε_z=4.10^{-12} m.rad) generating Xray from insertion devices installed on 5 m long straight sections. With β_x=36m and β_z=2.5m in the center of the high beta straight sections, the rms beam sizes at the BPMs located on both ends of the straight sections are σ_x= 380µm and σ_z= 14µm. The parasitic motion of the beam due to slow drifts or high frequency vibrations of the quadrupole support girders must be kept at low enough values to avoid spoiling this emittance figure. Two kinds of motions can be observed: very slow drifts and vibrations at 7Hz, 30 Hz and 60 Hz. The amplitude of these vibrations at the ends of the straight sections is 10 µm rms horizontally and 3µm rms vertically. The components of the correction scheme are shown in figure 1, all BPMs and steerers in place are used by the system.

System Layout

The design of our new system is based on the availability of the Libera Brilliance electronics and an associated "Communication Controller" developed at DLS [1] and using the Libera Rocket I/O ports. This allows the measurement and broadcast of the beam position at 224 locations with a very good resolution at a rate of 10 kHz. We are using the 96 corrector magnets embedded in the sextupole cores to steer the beam, and since the power supplies feeding these magnets are installed in four locations, this particular constraint sets the architecture and topology of this system. Therefore, the correction computation is placed close to the power supplies and spread over 8 processors, 2 per location.

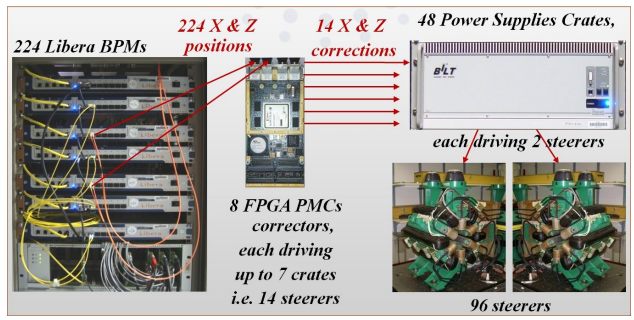


Figure 1: Fast Orbit Feedback components.

Upgrade of the Correctors Power Supplies

The present slow correctors are implemented by 3 pairs of auxiliary coils placed on the yoke of the sextupoles. Using the proper combination of currents in these 3 coil pairs we can produce any combination of vertical and horizontal kicks. The bandwidth of these correctors is affected by the eddy currents in the sextupole core and at the surface of the vacuum chamber since the vacuum chamber all over the storage ring is made of 2mm stainless steel. The inductance of these correctors is also quite large: 0.6H, however, it is possible to achieve, given the small amplitude of the high frequency currents needed for the fast correction, a small signal bandwidth of 500Hz thanks to a proper design of the power supplies. These power supplies are controlled through the Ethernet control system network and an additional trim setting can be added at a rate of 10KHz on 10% of the full dynamic range. This is done through a RS485 port used to input the data in a deterministic way and with a delay limited to 20µs. Among the features of the power supplies, a diagnostic on each of the 288 channels is available to check the proper functioning of the serial links.

Processing

For the processing, we selected a PMC module with fiber-optic transceivers and a Xilinx Virtex-5 FPGA. The code embedded in this FPGA has several functions:

1) Collect the data from the BPMs at 10 kHz with the Communication Controller

2) Obtain the parameters from the PCI

3) Process the corrections

4) Send the set-points to the power supplies

5) Send the set-points for data recording with the Communication Controller

The choice of performing the corrections inside the FPGA was driven by the fact that no real-time operating system was supported at the ESRF when we launched this development. One clear advantage of this solution is the efficiency of the FPGA: an interesting feature is the capacity to do parallel computing and also the fact that the time lost transferring the data is very limited since all the functions are performed inside the same component. One particularity of this development is the use of System Generator from Xilinx / The MathWorks to design the part dedicated to the corrections and sequencing of the system. The design can be checked with Matlab / Simulink before the VHDL code generation.

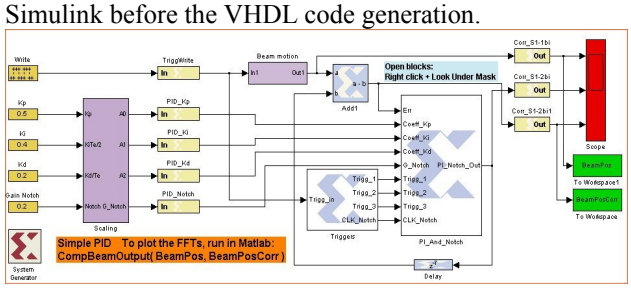


Figure 2: Corrector behaviour test with Simulink.

In order to check the behavior of the data processing at design stage, real data are introduced at the initialization of the simulation and processed with the components from Xilinx and then, output data can be transferred for analysis under Matlab. In the simulation of Fig.2, the power supplies, magnets and vacuum chambers are modeled as a simple delay of 700µs between the correction output and the reading of the beam position.

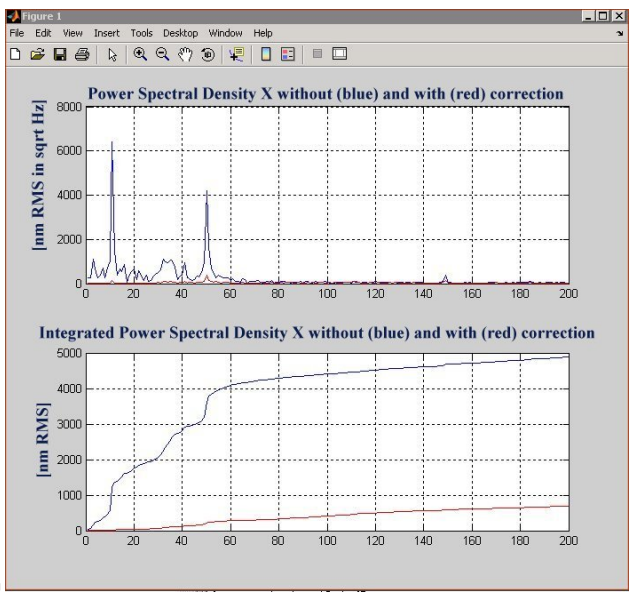


Figure 3: Theoretical results expected from the correctors.

Orbit Correction Algorithm

We derive the orbit correction from the BPM data using a correction matrix obtained from the inversion of the response matrix of the BPMs to each corrector. These response matrixes are inverted using the SVD method. We use 96 eigen vectors for the inversion of the response matrix. Before starting the 10 KHz correction loop, we will measure the average orbit and set the corrector currents in order to suppress the error measured on this average orbit; these DC values will be applied using the Ethernet input of the power supplies. We will then start the fast correction loop which will add an additional trim current to the DC current set initially. During the first tests of this fast loop, we used a PI algorithm with an additional 50 Hz notch filter aimed to improve the damping of the perturbation at the AC main supply frequency for the 10 KHz iteration of the values of the trim correction currents. Over long periods of operation, the average value of the trim currents may eventually drift up to significant values. In this case, this average current will be added to the setting of the Ethernet input of the power supply, and the average value of the fast trim currents will drop to zero. In this way, if the fast loop is stopped, setting the values of the trim currents to zero will only result in a very small orbit jump and we keep the whole dynamic range for the fast correction.

Diagnostic

In addition to the 8 feedback processor/power supply controller modules, one FPGA PMC board has been added; this board is fully dedicated to diagnostics, and is able to log up to 10s of position and correction data; this board is a duplication of the so called “sniffer” developed for the SOLEIL and DLS orbit control systems[2]. It is also used to detect errors in the data collection by counting the “BPM data not received”.

TESTS

Orbit Correction Test Set Up

All the correctors are now equipped with their new power supplies and connected to the 8 FPGA processors themselves connected to the 224 BPMs.

Tests Results

The commissioning of the whole system started in September 2011 and the results are as expected (Fig. 5) apart from the correction of the 50Hz which is missing and will be performed with an additional notch filter which has already been tested with a reduced setup [3].

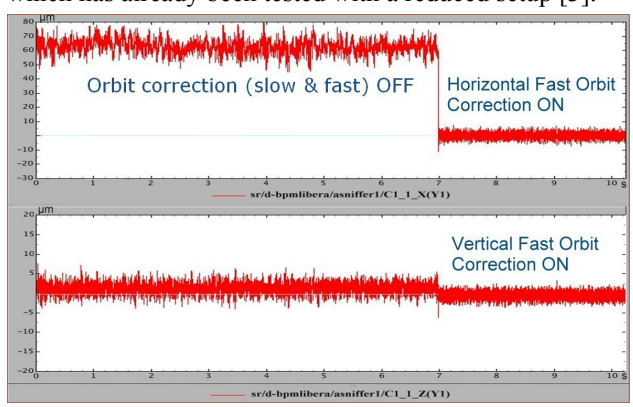


Figure 4: BPM signals with orbit correction OFF and ON.

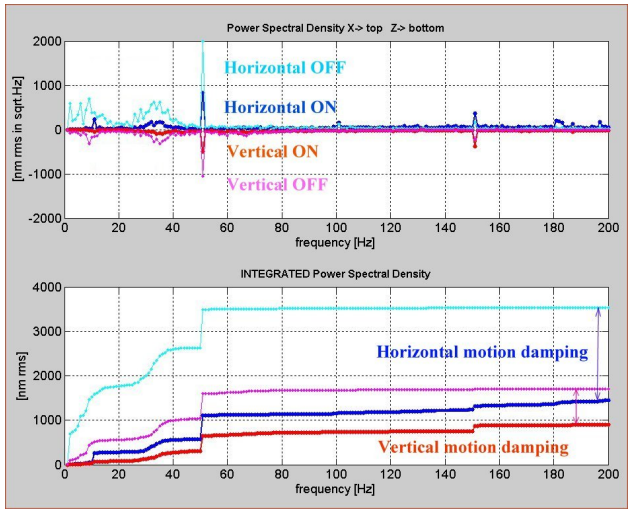


Figure 5: Beam motion averaged over 224 BPMs.

The correction bandwidth was set at 150Hz, and no weighting was applied on the damping time of the upper order eigen vectors of the SVD decomposition. We recorded the positions with the sniffer module. After the suppression of the remaining disturbance at 50 Hz, both horizontal and vertical motion will be reduced to less than 1µm RMS in a 200Hz bandwidth.

The plots of figure 5 show the reduction of the horizontal and vertical BPM signals. It is calculated from an average of the 224 positions.

We have also tested that starting from an orbit already set by a slow orbit correction, turning on or stopping the fast loop was not causing any significant orbit jumps.

Another test has been performed to assess the capability of the correction to drastically reduce the effect of the gap motion on the beam position. In fig. 6 we can see at first a record with no correction, the second part shows the result with a feed-forward control, the efficiency of which depends on the beam parameters, and lastly with the Fast Orbit Feedback "ON", the beam stays at its position within 1µm during the phase changes.

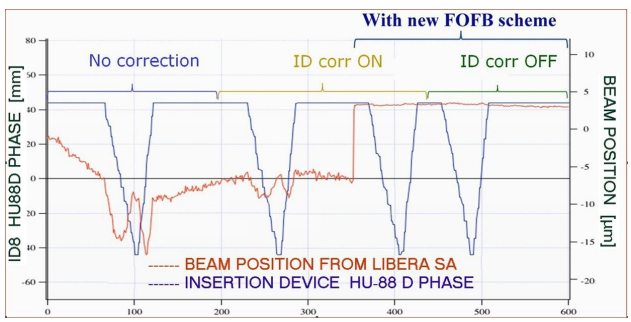


Figure 6: Correction of a beam displacement induced by an insertion device motion (Courtesy of J.Chavanne).

Other Measurements

The resolution of the 10 KHz FA data is 600nm in the range of currents that we store in operation. Such a resolution allows the measurement of SR parameters such as the measurement of the matrix of the response of the BPMs to the corrector current, or the analysis of the SR optics coupling with a very low excitation of the beam and a short acquisition time; applying excitation signals modulated by a sine signal at a well chosen frequency and a narrow bandwidth analysis of the beam response at this frequency, using the method tested at DLS [4], we checked that with a measurement time of 1s, a resolution of 6nm was achieved. We have used the sniffer to record the response of the BPMs to a 40 Hz modulated kick from a horizontal corrector over 1 second. The horizontal response amplitude is 5µm and the vertical response which is due to the coupling of the horizontal and vertical optics is very clean, though its amplitude is only 150nm.

CONCLUSION

We have tested the performance of this system and the damping of the orbit distortion that we achieved fulfils our expectation. There are some outstanding improvements to be made to the design of the correctors as well as to the control with the device servers. Then the next step will be to put this system in operation with a high level of reliability; this part requires special attention to the sequencing and diagnostic of the system. Concerning the development and commissioning, we are on schedule and we expect to put it into operation in spring 2012 at the end of the long shutdown planned for the construction work.

ACKNOWLEDGEMENTS

This project uses essential subsystems from Diamond Light Source and SOLEIL. We wish to thank G. Rehm, M. Abott and I. Uzun from DLS and N. Hubert from SOLEIL for the support that we got from them.

REFERENCES

[1] I. S. Uzun, "Initial Design of the Fast Orbit Feedback System for Diamond Light Source", ICALEPS'05.

[2] N. Hubert, SOLEIL Fast Orbit Feedback System, Proc 2nd Libera users meeting, April 2008, Barcelona.

[3] E. Plouviez et al, "The new fast orbit correction system of the ESRF Storage Ring", DIPAC 2011.

[4] G. Rehm et al, Measurement of Lattice Parameters Without Visible Disturbance to Users at Diamond Light Source, Proc BIW 10, Santa Fe, May 2010.

BEAM SYNCHRONOUS DATA ACQUISITION FOR SWISSFEL TEST INJECTOR

B. Kalantari, T. Korhonen, Paul Scherrer Institute, Villigen, Switzerland

Abstract

A 250 MeV injector facility at PSI has been constructed to study the scientific and technological challenges of the SwissFEL [1] project. Since in such pulsed machines in principle every beam can have different characteristics, due to varying machine parameters and/or conditions, it is very crucial to be able to acquire and distinguish control system data from one pulse to the next. In this paper we describe the technique we have developed to perform beam synchronous data acquisition at 100 Hz rate. This has been particularly challenging since it had to provide us with a reliable and real-time data acquisition method in a non real-time control system. We describe how this can be achieved by employing a powerful and flexible timing system with well defined interfaces to the control system.

INTRODUCTION

The 250-MeV Linear accelerator [2] at PSI has been constructed as a test bed facility to help design, development of concepts and proof of principles toward realization of the SwissFEL project. This injector test facility will be also used as the injector of the SwissFEL.

In general in operation of pulsed machines, e.g. FELs, it is very crucial to be able to study and diagnose the machine behaviour on pulse-to-pulse bases. Therefore it is required that, the data generated by the machine components is measured and collected at each pulse. Furthermore it is also required that the collected data at each pulse is clearly distinguished from those of another pulse. The reliable, pulse-to-pulse data acquisition requires a tight cooperation and well defined interface of timing system with the control and measurement systems. Considering the repetition rate of such machines, which is normally high (from several tens to few hundred Hz), the data acquisition and collection, will not be anymore a trivial task to do. Our injector currently produces beam at 10 Hz repetition rate. However, since there are several subsystems, e.g. Laser systems, which require 100 Hz operation at the same time we have to be able to cope with data acquisition at 100 Hz.

The pulse-to-pulse data acquisition across several locally separated control/measurement nodes is also known as Beam-Synchronous Data Acquisition to which we refer by BS-DAQ in the rest of the paper. In the following we first describe our BS-DAQ concept and then briefly describe the timing system features and control system interface which helped us to implement this concept.

BS-DAQ CONCEPT

The problem of BS-DAQ is in fact that of real-time data acquisition. Hence the first and the most important issue to address is, to acquire and collect data in a specified time interval. The time interval is specified by the machine repetition rate, said the other way, time interval between two consecutive machine pulses. So it is crucial that a measurement device participating in BS-DAQ demonstrates a deterministic behaviour. Hence, in our mechanism we assume that the measurement device has this important property. Closely related to this, is the issue of presentation of the measurement in the control system. In this regard we also assume that the measured data is transformed to a control system object well within the deadline determined by the repetition rate.

Local Buffering Concept

The heart of the BS-DAQ concept is the local buffering of the generated data at the Input/Output Controller (IOC) node. A similar concept has been also employed in LCLS [3]. The major reason for this approach is that in our control system the communication between IOC's is via an Ethernet-based protocol (EPICS Channel Access) which provides no guarantees in determinism of the data transmission delay (no upper bound on the delay). However since the IOC's demonstrate a real-time behaviour at the local level, we collect the acquired data locally at the beam or machine rate and then after finishing the acquisition the data collected in the buffers can be transferred via the communication network to the remote client application for offline analysis i.e. correlation studies.

Role of the Timing System

To make the local buffering work, we still need to provide a minimal but reliable, real-time communication across all locally separated IOC's which participate in a BS-DAQ run. This is because the measured data belonging to the same pulse has to be synchronously buffered at each IOC and it has to be possible to distinguish each buffered element by a *pulse ID* (pulse identification number). We use global event system of Micro Research Finland (MRF) [4] for our timing system. In the event system, an event generator in the central master timing IOC generates timing information and transmits them to all event receivers around the facility via optical links. The pulse ID or pulse marker is a unique, monotonically increasing number which is assigned to each machine pulse. It is generated in the central, master timing IOC and distributed via the timing system to all IOC's connected to the timing system (via

optical links). The pulse ID is received at each IOC via the event receiver and is transformed to a control system object where it can be used as an index of the acquired data at each pulse.

Role of the Control System

The control system at each IOC provides the means for control logic of the buffering mechanism. In our case all the logic, mechanisms and required software objects has been provided by the control system (EPICS) such that no special object type or low level driver had to be developed. The timing critical command/status data as mentioned is provided by the timing systems but finally presented as control system objects.

IMPLEMENTATION

In the implementation of the BS-DAQ mechanism, we have chosen right from the beginning to develop a generic control system software package so that the installation and usage for the control system engineers is a plug and play task which does not require any internal knowledge of the mechanism. In this regard we should emphasize that the EPICS toolkit had almost all the features that we required for implementation of the BS-DAQ.

Overview

The master timing IOC which houses the event generator is the central coordinator and synchronizer of the BS-DAQ. An EPICS program which is driven by the event systems interrupts determines, according to user specified parameters, at what pulses each IOC has to collect an EPICS channel. Then at each appropriate pulse, a command is sent synchronously to all IOC's via the timing system. When the receivers in each IOC receive the BS-DAQ commands, they find out if they have to copy value of the specified EPICS channel into the specified local buffer. The local buffers are realized by using a standard EPICS record type called "compress". Along with each data buffer there is also a pulse ID buffer to serve as the data index. A BS-DAQ run is initiated on-demand by the user for specified number of pulses. Typically the buffer length is selected such that at least 20 seconds interval can be covered. For example at 100 Hz repetition rate a buffer with 2000 element should be sufficient.

Control and Tweaking Knobs

A user (machine expert usually) who runs the BS-DAQ has some knobs to control. These are as the following:

- Number of acquisitions: this determines the total number of samples which will be collected in the local buffers at each IOC.
- Spacing: this parameter specifies the spacing between beam pulses which are sampled. For example zero spacing causes the data collection at each generated beam. A spacing of one, means samples at every second beam pulse will be collected.
- Defer cycles: this specifies how long to wait after each beam before buffering of each sample.
- Start/stop/abort/resume: these are the major control knobs to start, stop, resume or abort a BS-DAQ run.

Multi-user Operation

The BS-DAQ application is implemented such that it allows several users to simultaneously run their own independent BS-DAQ instance. Each user *leases* an acquisition *slot* and takes control over that until the acquisition is finished where the slot becomes *free* again (by the user). Currently six slots are supported and it can easily be expanded to more slots if required. In terms of logic, control and timing resources there is practically no limitation on increasing the number of acquisition slots. On the other hand number of machine experts that would use the application at the same time will not be a big number.

Dynamic Configuration

Most of the configuration parameters of the BS-DAQ can be specified at runtime. In particular the user can decide what data (EPICS channel) goes to what available buffer on an IOC. This makes the application very flexible since it is not necessary to specify measured data and the buffer which is going to be used in BS-DAQ at IOC initialization time, hence there is no need to restart the system. Furthermore, it allows better utilization of the IOC resources such as memory. It also leads to smaller and easier maintenance of the software package. Dynamic configuration, on the other hand, makes setting up an acquisition slot a rather complex task which involves several steps such as finding a free slot, slot leasing, finding free buffer, buffer assignments, etc. Currently the high level user application that retrieves and analyzes the collected data does this setup task. The high level software that our machine experts use for this purpose is Matlab.

First-Level Verifications

To provide a simple, first level cross-check in order to see if the BS-DAQ has done a successful run, we provided two verification methods. The first is to check if the data (EPICS channel) to be taken is not in the alarm state. The second is a check to see if there is any irregularities in the collected pulse ID's along with each collected data.

Support for Waveforms

The BS-DAQ is also able to collect the vector data (that has more than one element) at each sample in the same way as it does for scalar data. This was needed to collect the camera images or phase and amplitude waveforms in the LLRF systems. The BS-DAQ mechanism handles vectors data exactly in the same way as scalars. This was

achieved by introducing a parameter to BS-DAQ logic, applied to every data, as *sample size* which is 1 for scalar data and N for a vector with size N.

CONCLUSION

In this paper we have presented our approach to perform Beam-Synchronous Data Acquisition (BS-DAQ) in the SwissFEL test injector facility. The concept of local buffering was presented and our implementation was discussed in details.

REFERENCES

[1] http://www.psi.ch/swissfel/swissfel

[2] T. Schietinger, M.Aiba, B. Beutner, M. Dach, A. Falone, R. Ganter, R. Ischebeck, F. Le Pimpec, N. Milas, P. Narang, G.L. Orlandi, M. Pedrozzi, S. Reiche, C. Vicario, " FIRST COMMISSIONING EXPERIENCE AT THE SwissFEL INJECTOR TEST FACILITY", Proceedings of Linear Accelerator Conference LINAC2010, Tsukuba, Japan Paul Scherrer Institut, CH-5232 Villigen PSI, Switzerland

[3] J. Dusatko, S. Allison, M. Browne, P. Krejcik, "THE LCLS TIMNG EVENT SYSTEM", Proceedings of BIW10, Santa Fe, New Mexico, US

[4] http://www.mrf.fi/

DESIGN AND TEST OF A GIRDER CONTROL SYSTEM AT NSRRC

H.S. Wang, Y.L. Tsai, S.Y. Perng, M.L. Chen, K.H. Hsu, W.Y. Lai, T.C. Tseng, J.R. Chen
National Synchrotron Radiation Research Center, No. 101, Hsin-Ann Road, Hsinchu, 30076, Taiwan, R.O.C.

Abstract

A girder control system is proposed to quickly and precisely adjust the displacement and rotating angle of all girders in the storage ring with little manpower at the Taiwan Photon Source (TPS) project at National Synchrotron Research Center (NSRRC). In this control girder system, six motorized cam movers supporting a girder are driven on three pedestals to perform six-axis adjustments of a girder [1][2]. A tiltmeter monitors the pitch and roll of each girder; several touch sensors measure the relative displacement between consecutive girders. Moreover, a laser position sensitive detector (PSD) system [3] measuring the relative displacement between straight-section girders is included in this girder control system. Operator can use subroutines developed by MATLAB [4] to control every local girder control system via intranet. This paper presents details of design and tests of the girder control system.

INTRODUCTION

TPS girder control system consists of twenty four girder control systems and a girder-position computer. A girder-position computer grabs each girder's observation values indicating six degree of freedom, three degrees of freedom come from touch sensors; two degrees of freedom are from a tiltmeter and one degree of freedom from a PSD system. The system calculates all girder best positions by minimizing global girder position errors and the algorithm [5] is developing. Each girder control system receives six motorized cam mover angles determined by a girder adjustment algorithm from the girder-position computer via intranet. A girder control system driver drives a girder with six motors to adjust quickly and precisely the displacement and rotating angle of the girder. An adjustment cycle is finish. Locking systems are applied to fix all girders after the global girder position error achieves a good required precision for the whole storage ring. One twenty-fourth of the whole ring is shown as figure 1.

Figure 1: One twenty-fourth of the whole ring consists of three girders.

A GIRDER CONTROL SYSTEM

In order to achieve one micrometer-level displacement [6] and one microradian-level revolution of the rotation of a girder, a girder control system includes touch sensors provides resolution of thirty nanometers and two micrometer of accuracy, a tiltmeter provides resolution of one microradian and a PSD system has resolution of two micrometers at thirteen meters propagating distance every four hours. A girder control system synchronously controls three girders and each girder with six motors is driven co-ordinately to keep each girder's movement at minimum rotating variations.

The whole controller systems are adopted PXI platform and Microsoft Windows 7. PXI architecture is PC-based platform for control and measurement. The advantage of PXI is easy to design control system without other extra learning. The disadvantage is that the control system is unstable without good programming techniques and schemes. A girder control system is shown in figure 2. Eighteen motor drivers are arranged in the upper cabinet, PXI controller in the middle and custom electronic circuits for motors and encoders system in the bottom of the cabinet.

Figure 2: A girder control system.

A Cam Mover Control System

In order to provide good support of magnets and align the girder precisely and quickly, each girder is constructed with six cam movers on three pedestals to realize six-axis adjustments with automation. In addition to eighteen automatic motor baker control channels, each cam mover control system possesses eighteen stepper motor control channels and twelve critical signal stops. Those stops' quantity is adjusted channels depended by practical requirements. The control system provides driving eighteen motor synchronously with three girders coordinate movements.

In order to achieve the requirements, NI PXI-7813R [7] with Xilinx Virtex-2 is chosen as a customized motion card of the cam mover control system for the girder control system. The algorithm of a motion card is programmed by VHDL in addition to NI CLIP method. Applied to NI library, the girder controller handles the rotating angle of each motor in Microsoft Window 7 environment.

To reduce the entire installation time and human errors, customized circuits are designed for wiring installation. The customized circuits includes eighteen motor control channels, eighteen automatic motor baker control channels and thirty-two critical signal stops as figure 3.

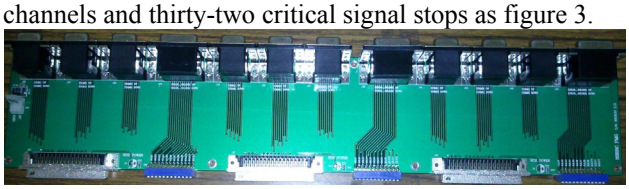


Figure 3: Eighteen motor control circuits for a cam mover system.

A Rotational Encoder and Touch Sensor Reading System

Rotational encoders and touch sensors provide absolute position with resolutions of 25-bit counts per rotation and 32-bis counts in twelve mm respectively. Sensors adopt Endat 2.2 protocol provided by Heidenhein as an interface to transit position data to sequential equipments.

In order to provide PXI encoder cards with higher channel quantity of Endat 2.2 protocol, NI PXI-7811R with Xilinx Virtex-2 is chosen as a customized encoder card. PXI-7811R does not possess differential circuits to be compatible to rotational encoders and touch sensors with Endat 2.2. The extra circuit is designed to be compatible to Endat 2.2 specs. A PXI-7811R grabs 18 rotational encoders and 18 touch sensors synchronously and the update rate achieves 10 kHz. It is similar to motion card of the cam mover system. The extra circuit board (figure 4) reduces installation time and human errors.

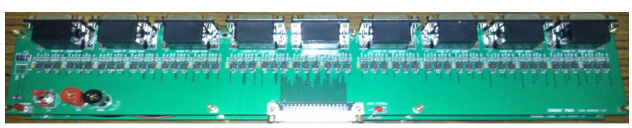


Figure 4: Eighteen differential circuits for encoders and touch sensors.

A Tiltmeter System

To measure the pitch and roll direction of a girder with one microradian resolution, Nivel 220 [8] is chosen as a tiltmeter. One twenty-fourth of the whole ring consists of three tiltmeters and those sensors connect together by RS-485 protocol. Before the installation or after any slight impact, a tiltmeter has to be calibrated.

A PSD System

The laser positioning system, a part of a girder auto-alignment scheme, will be installed on the girders located at both sides of each straight section of the storage ring. The system (figure 5) is composed of a laser and four sets of a position sensing device (PSD). The laser propagates thirteen meters along the girder and plays the role of a reference line of girders of the straight section. Based on the laser linear characteristics, the other girder can be adjusted and aligned by a cam mover according to PSD data. To achieve superior precision, the whole laser positioning system should be constructed stably. The precision of the laser positioning system can achieve two micrometer at thirteen meters propagating distance every four hours.

Each PSD provides a PCI DAQ card to detect laser position and power.

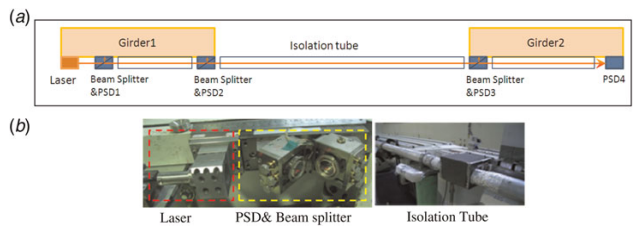


Figure 5: (a) Architecture of the laser positioning system. (b) Main portions of the laser positioning system.

A Locking System

After the global girder position error achieves to required precision for the whole storage ring, locking systems (figure 6) are applied to fix all girders. Three couples of locking systems installed on three pedestals between a girder and pedestals improve the resonance frequency. A locking system includes a wedge mechanism and electronic circuits. A couple of wedge mechanisms driven by DC motors push the wedges to reduce gaps between wedges and the girder to cause a clamping effort.

In order to control the wedge mechanism, a locking electronic circuit is designed to drive a couple of the wedge mechanisms synchronously. A DC motor is driven by PWM method and the motor rotating angle is monitored by the encoder installed at the end of the motor. The locking electronic circuit provides eighteen driven channels. Two channels can be controlled at once, and switching techniques help us to actuate eighteen DC motors. The motion card with PXI interface is programmed by VHDL with FPAG chip and also provides critical signal stops.

The locking system can provide 2000 kg of the maximum force at the condition that motor runs at 24V and 2A.

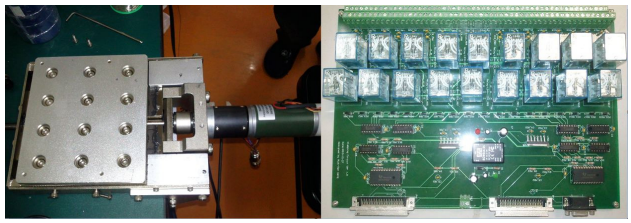


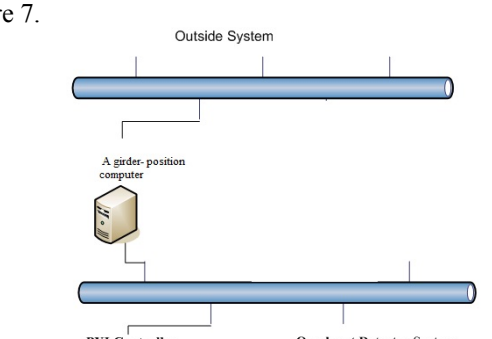
Figure 6: A wedge mechanism and electronic circuits.

Network and System Architectures

TPS girder control system consists of 24 girder control systems and a girder-position computer. To avoid virus risks, all control systems are in the individual network except a girder-position computer connecting internet for transiting sensors massage to Archive system.

Each girder control system receives six motorized cam mover rotating angles from the girder-position computer in the individual network. To increase the facility of control hardware, the basic subroutines developed by Matlab communicates to girder control systems. Users just study the algorithm for the adjustment without understanding hardware.

The girder control system provides the control of cam movers and the reading of encoders, touch sensors and tiltmeters. To increase system stability, all functions are developed to a small program with TCP/IP. Users read the sensors' data or control actuators via the individual network. The network and system architecture is shown as figure 7.



Name	Quantity	Desciption
PXI Computer	1	Compute the argrthm of girders
RS232 Port	1	Read data from Nivel 220
Encoder Card	4	Read Endata 2.2 signals of rotary and linear encoders
Digital I/O Card	2	For limit switches
Motion Card	1	Control Stepping Motor

Name	Quantity	Desciption
Sever	2	Operation System
Quadrant Detector card	2	Read the voltage of a quadrant detector

Total quantity of local side computers is 24

Figure 7: Network and system architectures.

TEST RESULTS

After stability tests, a girder system runs normally for more six months. Many Experiments are processing by girder control systems and the statuses of the systems are stable. The graphical user interface is shown as figure 8.

The locking system is applied to lock girders and the test results are presented as table 1. The deviation of a locked girder is less than 10 micrometers in the transverse. It is not controllable in other two directions and the deviation is larger than the transverse. The vertical and longitudinal movements of the girder are due to the locking force.

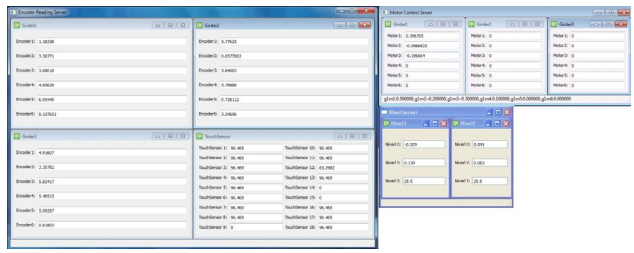


Figure 8: The graphical user interface of a girder control system.

Table 1: The Deviation of a Locked Girder

Deviation	Transverse	Vertical	Longitudinal
First	-3 μm	8μm	6μm
Second	-9μm	20μm	-17μm

CONCLUSION

A six-axis adjustments of a girder are developed automatically to adjust the displacement and rotating angle of all girders in the storage ring quickly and precisely with little manpower. The test results show that a girder control system adjusts girders quickly, precisely and stably. However, EPICS system will be adopted in Taiwan Photon Source. Therefore, our next step is to change the current system with EPICS.

REFERENCES

[1] T.C. Tseng and others, "A Precise 6-axis Girder System with Can Mover Mechanism," MEDSI, 2006
[2] T.C. Tseng and others, "Design and Prototype Testing of the Girder System for TPS," SRI, 2008.
[3] M.L. Chen and others, "Design Improvements and Tests of a Laser Positioning System for TPS Girder System", MEDSI, 2010
[4] http://www.mathworks.com
[5] W.Y. Lai and others, "Design and prototype tests of auto-alignment of a whole-ring girder", PAC09, 2009
[6] http://www.heidenhein.com.
[7] http://www.ni.com
[8] http://www.leica.com

BEAM SPILL STRUCTURE FEEDBACK TEST IN HIRFL-CSR*

R.S. Mao#, P. Li, L.Z. Ma, J.X. Wu, Y.J. Yuan, J.W. Xia, J.C. Yang,
T.C. Zhao, Z.Z. Zhou, IMP, Lanzhou, China

Abstract

The slow extraction beam from HIRFL-CSR is used in nuclear physics experiments and heavy ion therapy. 50Hz ripple and harmonics are observed in beam spill. To improve the spill structure, the first set of control system consisting of fast Q-magnet and feedback device based FPGA is developed and installed in 2010, and spill structure feedback test also has been started. The commissioning results with spill feedback system are presented in this paper.

INTRODUCTION

HIRFL-CSR [1] is the post-acceleration system of Heavy Ion Research Facility in Lanzhou (HIRFL), which consists of double cooling storage ring system(CSR) and a radioactive beam line (RIBLL2). The beam is accumulated, cooled and accelerated in the main ring (CSRm), and will be extracted [2] in fast extraction mode to experiment ring (CSRe) for internal-target experiments, or extracted in slow extraction (RF-knockout) mode for external-target experiments and cancer therapy.

To prevent the pileup events in particle detectors [3], and to improve the lateral dose distribution in the irradiation for heavy ion therapy [4], the ripple noises of beam spill should be suppressed. We developed a test set of the spill feedback system, which consists of two quadrupole magnets, commercial FPGA card and waveform generator.

SPILL CONTROL

In CSRm, the RF-knockout method is employed for slow extraction. Normally, the horizontal tune in CSRm is set to 3.662 by using normal quadruple pairs during slow extraction, and additional sextupole magnets are used to excite the third-order horizontal resonance. The centre frequency of RF–KO is about 1.666 times of the beam revolution frequency, and the span is 0.5%.

To improve the beam time structure, one commonly used method is RF-KO with amplitude modulation (AM) and frequency modulation (FM) [4-7], and the spill ripple can be suppressed by using fast Q magnets[8-11]. The FQ control signal should be the reverse phase of ripple noise [9], Figure 1 shows the example of FQ control signal. Figure 2 shows the block diagram of the spill feedback system in CSRm.

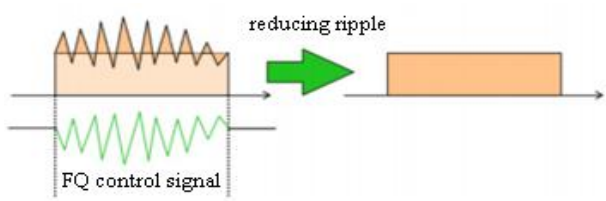


Figure 1: Spill control by FQ. [9]

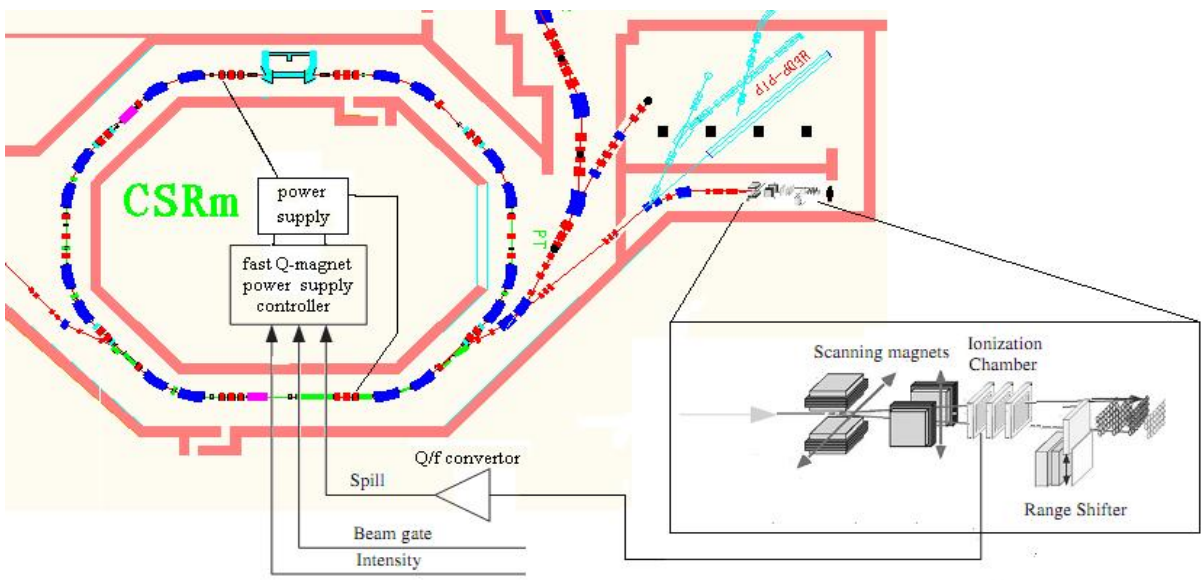


Figure 2: Block diagram of the spill feedback system.

*Work supported by NSFC…
#maorsh@impcas.ac.cn

The test set of spill control system in CSRm is based on the two methods mentioned above, system consisting of two fast Q-magnets, feedback device based FPGA and waveform generator. The beam current is measured by ionization chamber, and the output signal from Q/f convertor is TTL pulse[12], for example one pulse for 814 particles (200MeV/μ $^{12}C^{6+}$). The fast Q magnet power supply controller is (NI 7830R)[13], the parameters is shown in table 1, 1MS/s is enough for suppress the ripple lower than 500Hz. Figure 3 shows the block diagram of spill feedback processing, the maximum loop rate is 1MHz.

Table 1: Specification of NI 7830R

AO resolution	16 bits
AO update rate	1MS/s
max clock rate	40MHz
FPGA type	Vitex-2 Virtex-II 1M

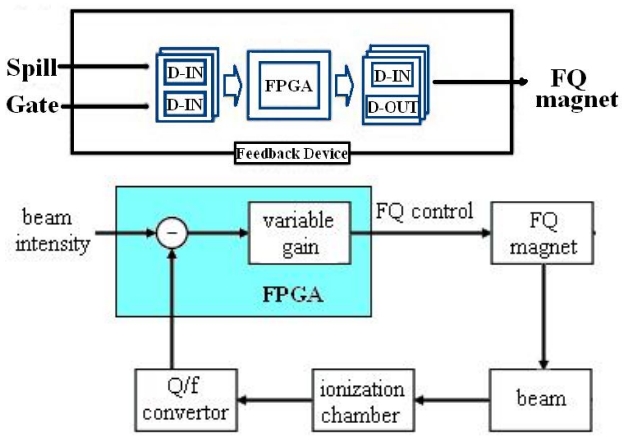


Figure 3: The spill feedback processing.

The best amplitude modulation curve [7] for RF-KO should like Fig. 4 curve(a), but the waveform generator TEK3252 [14] we have does not support real-time change of output amplitude, so we use curve(b) instead of curve(a), use software to change the output amplitude through GPIB port, refresh rate is about 35Hz.

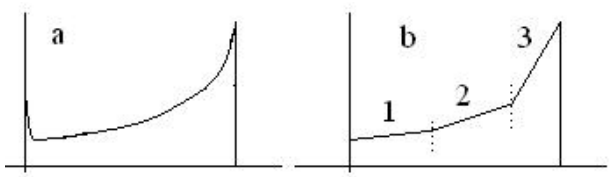


Figure 4: The AM curve for RF-KO, (a) best curve, (b) three lines.

There are two fast Q magnets symmetrically installed in CSRm, so the lattice will not change by using FQ(Fig. 5). The horizontal and vertical tune value will be change at same time because of FQ, but the change of the vertical tune value is acceptable. Table 2 shows the fast Q magnets specifications.

Figure 5: FQ and power supply.

Table 2: Fast Quadrupole Magnets Specifications

Core Material	0.5mm thick lamination steel
Bore Radius	85 mm
Magnet Length	0.30 m
Coil Turn Number	3
Field Gradient	0.2T/m@370A
Inductance	0.3mH
Resistance	4mΩ

BEAM COMMISSIONING

The commission result is shown in Figure 5. The beam in CSRm is 200MeV/μ $^{12}C^{6+}$. Normally, horizontal tune value in CSRm is set to 3.662, and RF-KO set constant RF power, Fig. 6(a) shows the beam spill. If we change the RF amplitude during extraction, beam spill and FFT result is shown in Fig.6 (b).

Because the FQ magnet power supply is still under construction, we install one BUMP magnet power supply for temporary use. This power supply is single polarity, fall time (200A-0A) is 50μs, and rise time (0A-200A) is 200μs. Before the feedback system is turn on, horizontal tune value is set to 3.660-3.661, and we find that the beam cannot be extracted without feedback system turn on under this condition. Turn on the feedback system, and carefully change the parameters set in FPGA program, finally the beam spill become flat, the best result is shown in Fig. 6(c), ripple of 50Hz and its harmonic lower than several hundreds hertz is reduced.

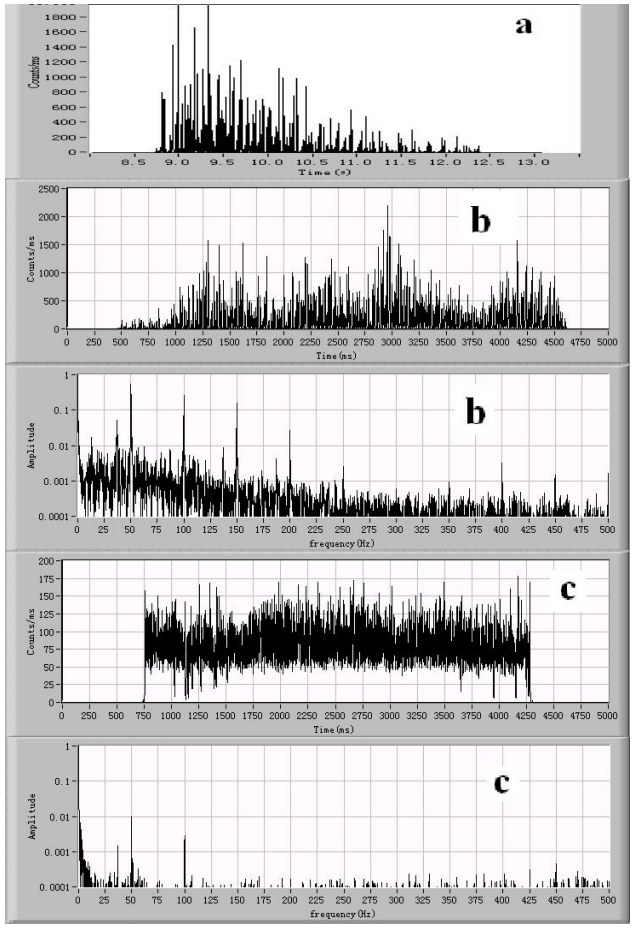


Figure 6: Beam spill structure and FFT results. (a) feedback off and RF-KO with constant RF power, (b) feedback off and RF-KO with amplitude changing, (c) feedback on and RF-KO with amplitude changing. Sample rate is 1KS/s. (200MeV/μ $^{12}C^{6+}$).

CONCLUSION AND NEXT STEPS OF DEVELOPMENT

- We finished developing the test set of spill feedback system, which consists of two fast Q-magnets, one feedback device based on FPGA, and one RF exciter based on waveform generator, and carried out the experiments in CSRm. The feedback system improved the spill structure, ripple of 50Hz and its harmonic is reduced.
- It's difficult to commission by using the single polarity power supply, so the bipolar power supply is really needed.
- The core material of fast Q magnets is 0.5mm thick lamination steel, now we begin to construct new one with 0.2mm lamination steel.

REFERENCES

[1] J. Xia, et al., "The heavy ion cooler-storage-ring project (HIRFL-CSR) at Lanzhou", Nucl. Instrum. Methods, A488 (2002)11-25
[2] Y. J. Yuan, et al., "Design of slow extraction for HIRFL-CSR", APAC01, Beijing.
[3] A. Kiyomichi, et al., "Beam spill control for the J-PARC slow extraction", IPAC'10, Kyoto, Japan.
[4] T. Furukawa, et al., "Progress of RF-knockout extraction for the ion therapy", EPAC2002, Paris, France.
[5] K. Noda, et al., "Advanced RF-KO slow-extraction method for the reduction of spill ripple", Nucl. Instrum. Methods, A 492(2002)253-263.
[6] S. Sato, et al., "Dynamic intensity control system with RF-knockout sloe-extraction in the HIMAC synchrotron", Nucl. Instrum. Methods, A 574(2007)226-231.
[7] T. Furukawa, et al., "Intensity control in RF-knockout extraction for scanning irradiation", Nucl. Instrum. Methods, B 240 (2005)32-35
[8] I. Marrneris, et al., "AGS slow extracted beam improvement", PAC97, Vancouver, BC, Canada.
[9] S. Onuma, et al., "The spill feedback control unit for the J-PARC slow extraction", IPAC'10, Kyoto, Japan.
[10] T. Nakanishi, et al., "Slow beam extraction method using a fast Q-magnet assisted by RF-knockout", Nucl. Instrum. Methods, A 533(2005)400-406.
[11] H. Nakagawa, et al., "Development of a signal processing board for spill digital servo system for proton synchrotron", ICSLEPCS07, Knoxville, Tennessee, USA.
[12] Su Hong, et al., "A design of the supervision circuit for heavy-ion beam density", Nuclear Electronics & Detection Technology, Sep. 2009, Vol.29, No.5.
[13] www.ni.com
[14] www.tek.com

ARCHITECTURE AND CONTROL OF THE FAST ORBIT CORRECTION FOR THE ESRF STORAGE RING

F. Epaud, Jean-Marc Koch, Eric Plouviez, ESRF , Grenoble, France

Abstract

Two years ago, the electronics of all the 224 Beam Position Monitors (BPM) of the ESRF Storage Ring were replaced by the commercial Libera Brilliance units to drastically improve the speed and position resolution of the orbit measurement. Also, at the start of this year, all the 96 power supplies that drive the orbit steerers have been replaced by new units that now cover a full DC-AC range up to 200Hz [1, 2].

We are now working on the replacement of the previous Fast Orbit Correction Feedback system. This new architecture will also use the 224 Libera Brilliance units and in particular the 10 KHz optical links handled by the Diamond Communication Controller (DCC) which has now been integrated within the Libera FPGA as a standard option. The 224 Liberas are connected together with the optical links to form a redundant network where the data are broadcast and are received by all nodes within 45 µS. The 4 corrections stations are based on FPGA cards (2 per station) also connected to the FOFB network as additional nodes and using the same DCC firmware on one side and are connected to the steerers power supplies using RS485 electronics standard on the other side. Finally two extra nodes have been added to collect data for diagnostics and to give BPMs positions to the beamlines at high rate. This paper presents the network architecture and the control software to operate this new equipment.

MOTIVATION

The present Fast Orbit Correction installed in 2004 and using only 32 BPMs, 32 correctors in the horizontal plane and 16 correctors in the vertical plane, is rapidly aging and is less and less reliable. Also its' Electronics based on VME-DSP71 cards using C40-DSP and developed at the ESRF, PXI-Sundance C67-DSP cards and Windows 2000 operating systems is now obsolete. Also, we have to rely on very few spares to operate this equipment. It is working at 4.4 KHz sample frequency and corrects the beam positions in a bandwidth from 0.05Hz to 150Hz.

Also since 2 years the 224 Liberas Brilliance devices [3] connected to the 224 BPM heads and also the 48 BILT power supplies connected to the 96 steerers are in operation to perform the slow orbit correction every 30 seconds. The correction loop is performed by software using TANGO control system and several device servers reading positions data from the Liberas, calculating the corrections and writing them to the steerer power supplies over Ethernet and using standard TCP/IP protocol.

These two systems have recently being coupled [4].

Since the beginning of the slow orbit correction refurbishment we have also kept in mind the Fast Orbit Feedback upgrade and decided to use the continuous 10 KHz positions data stream delivered by the Liberas over the four 1Gbits/sec optical Links. We have also designed the BILT steerers power supplies in order to be able to apply fast correction over a RS485 serial line in addition to the slow correction coming from the Ethernet.

NETWORK ARCHITECTURE

Liberas Ring

When we bought our 224 Liberas, Instruments Technologies, the Libera manufacturer, proposed to use their standard "Gbits Ethernet protocol", but for several reasons we do prefer to use the Communication Controller protocol (DCC) developed by Diamond Light Source and already in use at Diamond and Soleil. Therefore for warranties, support and to not have to modify the Libera's embedded FPGA program, we asked to Instruments Technologies to integrate this protocol as an option of the Libera.

This add-on allowed us to connect our 224 Liberas (32 cells * 7 BPMs) all together using our standard Ethernet infrastructure. At the ESRF, the 32 cells are wired with a 12 pairs optical fibbers going to a central point behind the control room. Only one or two pairs are used for the Ethernet network and the rest is available for other purpose. Therefore we used 4 pairs to build our Liberas Fast network (4 pairs are necessary for redundancy).

The 7 Liberas of each cell are connected together with copper cables to form a primary ring then two of these Liberas are connected to the cell N-1 and cell N+1 via the optical fibber. Also two other Liberas are connected to cell N-8 and cell N+8 to make redundancy.

We had also to develop a synchronisation card which allows sending the required pulses to synchronise all the Liberas and to start the DCC, following a strict timing protocol. When the Liberas have been synchronised, the time to refresh all the X and Z positions is around 45 µs.

We have also developed tools to ease the commissioning of the Fast network, which will also be used latter during the USM (Users Service Mode) to survey this equipment. This software uses a TANGO device server which collects connections status on all the Liberas and displays them on a Java application. The Figure 1 below shows the status of the network with some faults on cell 26 which are highlighted in the window on the top to better determine the origin of the problem.

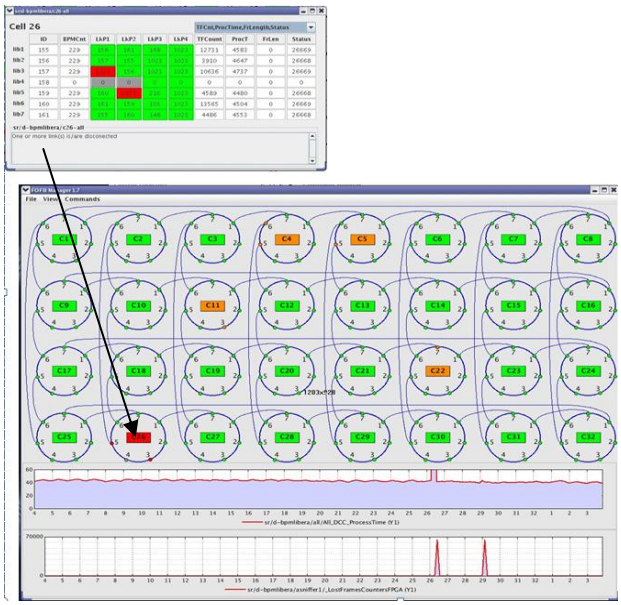


Figure 1: Java application which controls and surveys the Liberas Fast network status. Here is shown a problem with the Libera 4 of cell 26, which is not running and has lost the connection with its neighbours.

Sniffer

Even if the Liberas Fast network is running well, we need also to be able to collect the positions data. Therefore we use a PMC-FPGA-03 mezzanine module from Curtiss Wright which is connected on a PCI carrier. This module has the same optical coupler than the Liberas and allowed us to integrate the Communication Controller FPGA program to sample the data in real-time (@ 10 KHz) and sends them to a CPU with DMA cycle over PCI every 4096 frames (every 400 milli-seconds).

This card is plugged in a rackable PC running Windows XP. We finally have installed two sniffer systems: 1) To collect positions data used by a server which calculates the positions, angles, FFT and RMS in the middle of the straight sections and make them available to the beamlines, 2) To carry-on the commissioning of the whole FOFB system.

These two systems are two extra nodes of the Fast network.

Corrections Stations

At the ESRF, the steerers magnets of 8 cells are controlled by power supplies located at 4 areas around the storage ring, therefore we had to fit with this architecture already in place and had to install the corrections stations in these areas. We decided to use a card similar to the one already used by the sniffers, but more powerful and more up-to-date, the XMC-FPGA05F mezzanine module from Curtiss Wright [5]. We also used 4 Windows XP rackable PCI crates, but the FPGA simulation demonstrated that one card to handle 24 steerers was not enough and we had to use two cards per crates. This was also justified by the lack of place to connect the steerers power supplies interfaces.

Finally, each card can handle up to 14 steerers on both planes. Once again, these cards are connected to the Libera Fast network as 8 extra nodes. They collect positions data within 45 µs out of the 100 µs available, leaving 65 µs to calculate the corrections according to the correction matrix and the PID coefficients. It send also the corrections both to the steerer power supplies via RS485 and to the Fast network for debugging purpose (correction data are multiplexed at this point).

Synchronisation Mechanism

There is actually no hardware synchronisation mechanism to synchronise the start of the 8 cards of the 4 stations. They are started using TANGO group calls meaning that the jitter can be up to a few milli-seconds and more in case of network overload. This is acceptable but nevertheless we envisage to add such a mechanism.

SOFTWARE ARCHITECTURE

In fact, as far as the positions acquisition , the corrections calculation and the communication with steerer power supplies are made within the FPGA program, the control software using TANGO device servers is quite simple and is limited to provide the parameters to the FPGA cards and to survey the state of the 8 cards (See Figure 2).

FOFBCorrectionStation

This device server runs locally on each of the 4 Windows XP stations and manages two devices (one per card). One device/card can handle up to 14 steerers in both planes. Thanks to the 'fusionXP' driver, it communicates using a memory area on the PCI bus seen also from the FPGA. This area allows giving the PID, corrections and stimulus coefficients to the correction firmware. It also allows starting and stopping the correction and/or the stimulus and finally handles the errors coming from the FPGA.

FOFBCorrectionGlobal

On top of the 4 FOFBCorrectionStation device servers is the FOFBCorrectionGlobal device server allowing to manage the whole system as is there was only one correction station. This server manages two devices, one in both plane and has the knowledge of the architecture behind it. In particular it knows which steerers are managed by which stations and therefore can dispatch the corrections vectors for each steerer according to the correction Matrix provided by higher level application (matlab at the moment). Also it manages the errors of the 4 stations and can decide to stop the correction in case of problem.

FOFBSniffer

This device server handles continuously the DMA cycles coming from the FPGA via a FIFO. It collects the data and insert them into a circular buffer having a depth of 10 seconds. The clients can pick-up data on this buffer and can read BPM history. The recording can also be

frozen in order to not lose the synchronisation when several BPMs need to be read. Also the whole BPMs positions and the whole corrections values can also be read on a defined depth by one command to keep the synchronisation between data.

Applications, Commissioning

We are currently commissioning the whole system and had to develop some tools (as the LABView application below) in order to treat the huge flow of data. We also use a lot 'jive', the TANGO generic tool, to verify that the parameters coming from the higher application are dispatched on the right station and at the right place, for the right steerer. For example, this helped to discover that our steerers numbering was not the same everywhere and that the number one was the first steerer of the cell 1 for wiring aspects but was steerer 1 of the cell 4 for physics and computing aspects. One should know that at the ESRF, the first cell after the extraction to the Storage Ring is the cell number 4.

Also, Matlab is widely used by the diagnostics experts, especially this software is perfectly adapted to calculate and invert the correction matrix. Latter, an application for the Control room will be developed, may be in matlab like the previous system or in Java.

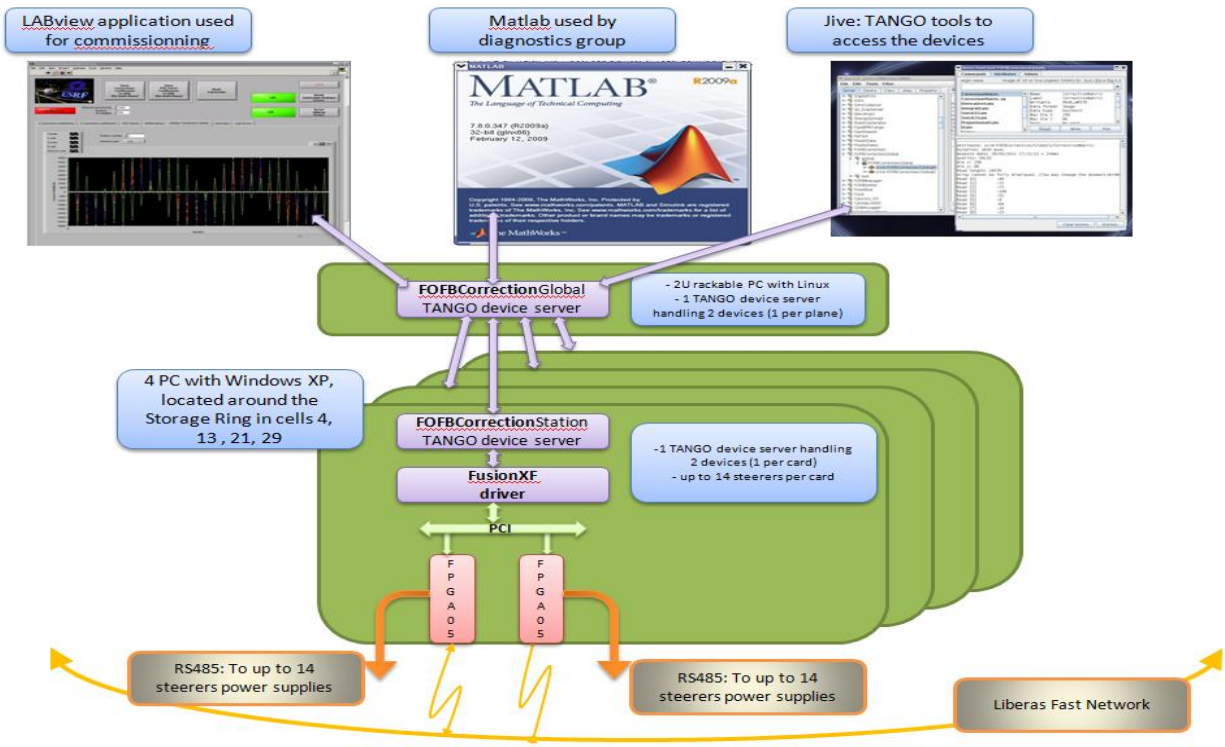


Figure 2: Software architecture of the Fast Orbit Feedback based on Liberas.

RESULTS

Very few MDT time has been allocated to the commissioning of this new equipment, therefore we are trying to make as much as possible during the USM and had to prepare carefully our MDT programs. We have also suffered from some bugs and human mistakes which provoke some beam loss or unexplained beam motion during the USM. Recently, we succeed to close the loop and to correct the beam positions in both planes [7].

REFERENCES

[1] K.B.Scheidt, F.Epaud, ESRF, Grenoble, Installation and Commissioning of a complete upgrade of the BPM system for the ESRF Storage Ring, MOPD05, DIPAC-09, Basel,

[2] F.Epaud, ESRF, Grenoble, ESRF's new beam position system for the storage ring using Libera Brilliance devices, THP004, ICALECS-09, Kobe

[3] Instrumentation Technologies. http://www.i-tech.si

[4] E.Plouviez, L.Farvacque, JM.Koch, JL.Pons, F.Uberto, ESRF, Grenoble, Improvement of the fast orbit correction on the ESRF storage ring, TUPD05, DIPAC-09, Basel.

[5] Curtiss Wright Controls http://www.cwcembedded.com/

[6] E.Plouviez, F.Epaud, JM.Koch, KB.Scheidt, ESRF, Grenoble, The New Fast Orbit Correction System of the ESRF Storage Ring, MOPD74, DIPAC-2011, Hamburg.

[7] J.M. Koch, F. Epaud, E. Plouviez, K.B. Scheidt, ESRF, Grenoble, Fast Orbit Correction for the ESRF Storage Ring, ICALEPCS-2011, Grenoble.

DIAGNOSTICS CONTROL REQUIREMENTS AND APPLICATIONS AT NSLS-II*

Yong Hu#, Leo Bob Dalesio, Kiman Ha, Om Singh, BNL, NSLS-II, NY 11973, U.S.A

Abstract

To measure various beam parameters such as beam position, beam size, circulating current, beam emittance, etc., a variety of diagnostic monitors will be deployed at NSLS-II. The Diagnostics Group and the Controls Group are working together on control requirements for the beam monitors. The requirements are originated from and determined by accelerator physics. An attempt of analyzing and translating physics needs into control requirements is made. The basic functionalities and applications of diagnostics controls are also presented.

INTRODUCTION

State-of-the-art beam diagnostics and control systems are required for a smooth and rapid commissioning and for productive and successful operation of the NSLS-II storage ring. The NSLS-II beam diagnostics and control system is designed to monitor the electron beam of NSLS-II accelerator complex. The beam quality is measured by a variety of parameters such as bunch charge, bunch structure (filling pattern), beam position/orbit, beam size/profile, energy & energy spread, circulating beam current, tunes, beam emittance, bunch length and beam losses.

A correct measurement of beam parameters depends on the effective combinations of a variety of beam monitors, control and data acquisitions (DAQ) and high level physics applications. Figure 1 shows the relationship between these systems.

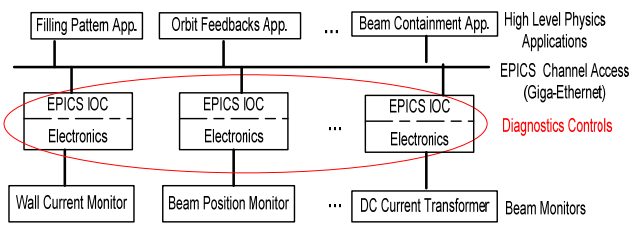


Fig. 1: NSLS-II Beam Diagnostics & Control Systems.

The following beam parameters will be monitored during regular operations:

- closed orbit (accuracy better than 10% of beam size);
- working point (tune for both planes with 10^{-4} resolution);
- circulating current (0.1% accuracy) and beam lifetime (1% accuracy);
- injection efficiency;
- filling pattern (1% of maximal bunch charge);
- emittance for both planes (10% relative accuracy);
- energy spread;
- individual bunch length (2 psec resolution);
- position of the photon beam for the insertion devices;
- coherent bunch instabilities;
- distribution of beam losses around the ring;

CONTROLS REQUIREMENTS

To measure various beam parameters, a variety of diagnostic monitors will be deployed in NSLS-II. The Diagnostics Group and the Controls Group are working together on controls requirements for these beam monitors. These Requirements are determined by accelerator physics. According to NSLS-II PDR [1], the following beam parameters will be monitored during storage ring regular operations. An attempt of analyzing and translating physics needs into controls requirements is made.

Table 1 lists the beam monitors associated with beam parameters and summaries of controls requirement.

Table 1: Diagnostics Controls Requirements

Beam Parameter	Beam Monitor	Controls Requirements
Fill Pattern	WCM, FCT, BPM button	sampling rate: 4GS/s; resolution: 8-bit; IOC rate: 10Hz
Profile/Position	Flag	Binary control for pneumatic actuator CCD: 1620*1220@15fps, IOC@10Hz
Position/Orbit	BPM	Single Pass Resolution: 30um rms 1um rms@10KHz; 0.3um rms@10Hz
Bunch Charge	Bergoz ICT & BCM	20KS/s with 16-bit; IOC@10Hz
Beam Current	Bergoz DCCT	20KS/s with 18-bit;1Hz for injection efficiency calculation;
Bunch Length	Streak Camera	Windows software by vendor
Tunes	Striplines	Ethernet-based network analyzer
Emittance & Energy Spread	Pinhole Camera	Stepper motor control with readback; CCD: 1620*1220@15fps, IOC@1Hz
Beam Stability	Spectrum Analyzer	Ethernet-based instrument control

Closed Orbit

Requirement: accuracy better than 10% of beam size. The beam position and closed orbit is measured by BPM

* Work performed under auspices of the U.S. Department of Energy
# yhu@bnl.gov

(Beam Position Monitor). The smallest beam size is expected to be 3.1 um at short ID (Insertion Device) location. So, the BPM pickup buttons and associated electronics should provide position measurement resolution (RMS noise) at 0.3um (10% of beam size) for long-term orbit drift which can be compensated by slow orbit feedback based on 10 Hz Slow Acquisition (SA) BPM data. Additionally, NSLS-II BPM system will provide 10 KHz Fast Acquisition (FA) data for fast orbit feedback (FOFB) as well as turn-by-turn (TBT) data and ADC raw data for physics studies and BPM system debugging. These applications require less position resolution, usually at tens of microns.

BPM associated with its electronics/receiver, is the key diagnostic instrument. Diagnostics Group and Controls Group are putting much effort into it. NSLS-II BPM receiver [2] provides different data flows (117MHz ADC raw data, 379 KHz TBT, 10 KHz FA and 10Hz SA) for different usages.

Working Points

Requirement: both planes with 10^{-4} resolution. There're several methods to measure tunes (the fractional par). Most of them need pickup BPM and excitation stripline. One common method is based on network or spectrum analyzer. NSLS-II revolution frequency is 378.7 KHz and the tunes are expected to be 32.35/16.28. 10^{-4} frequency resolution means ~10Hz (0.28 * 379 KHz ~100 KHz) scanning step for the analyzer. Another is to utilize (FFT) BPM turn-by-turn data to measure tunes. 10-4 resolution means that at least 5120 (1/ (2N) < =10-4, N>=5000, N = 5*1024) TBT samples are needed for FFT.

Circulating Current and Beam Lifetime

Requirement: 0.1% accuracy for circulating current and 1% accuracy for beam lifetime. This is measured by DCCT and associated electronics. Bergoz NPCT with its analog electronics can provide +/-0.1% accuracy. The NPCT has 10 KHz nominal bandwidth. Large bandwidth gives more noise in the measurement so that filtering it to 500 Hz is always a good practice. In this case, one digitizer with 1KS/s sampling rate should be sufficient. The required resolution for digitizer is determined by the requirement on accuracy of beam lifetime measurement: 2% for 20 mA with 60-hour lifetime and 1 minute measurement interval. 18-bit ADC seems adequate for all these applications.

Injection Efficiency

This is done by comparisons between the charge measured by ICTs at transport lines and that measured by DCCTs at Booster and Storage Ring.

Filling Pattern

Requirement: 20% bunch-to-bunch charge variation. Filling pattern is measured by high-bandwidth (>500MHz) diagnostics monitors such as WCM and FCT. The pulse width of the output signal from Bergoz FCT is about 1 ns. Required 20% means less than 8-bit. So, high-speed digitizer with 2GHz bandwidth, 5GS/s sampling rate and 8-bit resolution should be sufficient for fill pattern monitoring.

Emittance

Requirement: both planes with 10% relative accuracy. emittance is not directly measured by diagnostics. It's calculated from β-function value (assumed to be a constant at the dispersion free location) and beam size (measured by one pinhole CCD camera at one diagnostics beamline). 10% relative accuracy should be achievable by well-designed pinhole optics and high-resolution (1620*1220) digital camera.

Energy Spread

It's also calculated from beam size which is measured by one pinhole CCD camera at another diagnostics beamline assuming the emittance is constant and has been measured.

Individual Bunch Length

Requirement: 2 psec resolution. Bunch length is measured by streak camera which can provide 2 ps resolution.

Position of the Photon Beam

This is measured by X-ray BPM and associated electronics (current-to-voltage converter and digitizer). Additionally, the blades of XBPM are moved and positioned by stepper motors.

FUNCTIONALITIES AND APPLICATIONS

The basic functionalities of diagnostics controls can be summarized as:

- Measurement of various beam parameters (~10) via a variety of beam monitors (~16).
- Acquirement and processing of the signals from beam monitors via different electronics and EPICS IOCs.
- Provision of the processed data as EPICS PVs for high level physics applications.
- Support of Top-off operation by providing filling pattern measurement to meet the requirements of initial filling storage ring from zero to full charge at 10Hz for Linac injection and at 1(or 2)Hz for Booster injection, as well as 1-minute top-off cycle after filling up.

From the point of view of controls and applications, diagnostics and controls systems can be classified into the following groups, as shown in Figure 2:

- BPM subsystem for orbit feedbacks, lattice measuring, etc;
- Filling pattern measurements based on WCM, FCT, stripline/synchrotron light with photo-diode;
- Loss Control and Monitoring subsystem as well as injection efficiency involving ICT, DCCT, BLM and scraper;

- Camera-based diagnostics such as screen/flag, pinhole system, streak camera, and synchrotron light monitor (SLM);
- Network/Spectrum analyzer-based tune measurement and beam stability monitoring;

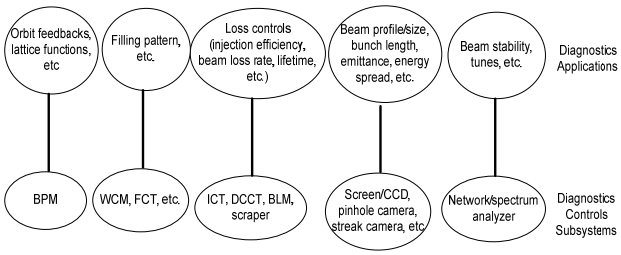


Fig. 2: Diagnostics Control Subsystems and Applications.

Orbit Feedbacks

BPM is the key diagnostics subsystem. BPM data is the source for NSLS-II storage ring fast orbit feedback (FOFB) and slow orbit feedback (SOFB). Orbit feedbacks are made of several parts including BPM pickup button, BPM electronics receiver, global BPMs data communication/exchange links, computing nodes for feedback algorithms, etc.

Diagnostics control system focuses on BPM receivers (Libera Brilliance or in-house electronics). It will provide 10 KHz fast acquisition (FA) data for FOFB over fast communication links (RocketIO-based Gigabit-Ethernet) and 10Hz slow acquisition (SA) data for SOFB over EPICS Channel Access (CA) network. Both FA and SA data from each BPM will be globally synchronized.

Besides orbit feedbacks application, BPM can be used for measuring turn-by-turn dynamics, chromaticity, dispersion, lattice functions, etc.

Filling Pattern Measurement

BPM Thanks to the high bandwidth of WCM (~3GHz), FCT (~1.7GHz), stripline (>1GHz), they can be used to observe individual bunch shape from the multi-bunch trains (80~150 bunches, 500MHz). So, WCM, FCT and stripline are chosen to measure filling pattern which is required for NSLS-II top-off operation.

The diagnostics controls will provide the data acquisition and process systems with over 1GS/s sampling rate and 8-bit resolution to observe the shape of each bunch whose duration is 2ns (500MHz) and calculate the charge of each bunch.

Loss Control & Monitoring

ICT (by Bergoz, coupled with BCM), DCCT (by Bergoz), BLM and scraper will be deployed in loss control and monitoring (LCM) which is designed for monitoring and controlling radiation losses in NSLS-II. LCM should be interlocked to top-off injection. The bunch charge losses (injection efficiency) will be monitored by ICT and DCCT. BLMs will be used to confirm beam loss locations around the ring. And scrapers can be used to intercept beam and control beam losses.

The diagnostics control system will digitize the DC voltage outputs from ICT&BCM, DCCT and then calculate the injection efficiency, beam losses rate, beam lifetime, etc.

Camera-based Diagnostics Applications

Various cameras will be used in beam diagnostics for measuring beam profile, beam size, emittance, energy spread, bunch length, etc. Screens (flags) coupled with CCD cameras placed over the whole NSLS-II accelerators are very useful to trace beam position and observe beam profile during machine commissioning. Synchrotron-light-based measurements in the Storage Ring diagnostics beamlines will be conducted on streak camera to measure bunch length and on pinhole camera to measure beam size, transverse emittances, energy spread, etc.

To standardize CCD camera controls, the digital cameras used in flags and pinhole-systems will be purchased from the same manufacture and have the same control/communication interface. Gigabit-Ethernet interface is preferable to FireWire (1394b) in terms of bandwidth, cabling, anti-EMI, etc. Prosilica Gig-E CCD camera is under evaluation. The diagnostics control system will acquire the digital image from CCD camera and then send the raw data (pixels) to MatLab-based high level application for image analyzing and processing(background subtraction, Gaussian fit, calibration, etc).

The streak camera system itself contains control & data acquisition software. It only interfaces to timing system. .

Network/Spectrum Analyzer-based Diagnostics

Tune monitor is used to measure transverse tunes by two strip-lines (signal pickup and source excitation) and one network analyzer. Beam stability monitor will observe spectrum of beam motion using one pickup strip-line and one real time spectrum analyzer.

Modern network/spectrum analyzers are usually equipped with Ethernet/GPIB interfaces and Windows XP operating system. For diagnostics controls, they can be characterized as Ethernet-based instrument control.

CONCLUSIONS

Collaboration between NSLS-II Diagnostics Group, Controls Group and Physics Group has been well established from very beginning of our project. Beam diagnostics requirements for controls are well understood.

REFERENCES

[1] NSLS-II Preliminary Design Report", http://www.bnl.gov/nsls2/project/PDR.

[2] Yong Hu, et al., "BPM Inputs to Physics Applications at NSLS-II", Proceedings of PAC 2011, New York, NY, US.

ELECTRO OPTICAL BEAM DIAGNOSTICS SYSTEM AND ITS CONTROL AT PSI

P. Chevtsov#, F. Mueller, P. Peier, V. Schlott, D. Treyer, PSI, Villigen, Switzerland
B. Steffen, DESY, Hamburg, Germany

Abstract

Electro Optical (EO) techniques are very promising non-invasive methods for measuring extremely short (in a sub-picosecond range) electron bunches. A prototype of an EO Bunch Length Monitoring System (BLMS) for the future SwissFEL facility [1] is created at the Paul Scherrer Institute (PSI). The core of this system is an advanced fiber laser unit with pulse generating, phase locking and synchronization electronics. The system is integrated into the EPICS based PSI controls, which significantly simplifies its operations. The paper presents the main components of the BLMS and its performance.

INTRODUCTION

Free Electron Lasers (FEL) are very powerful tunable light sources for a wide spectral range with excellent coherence. One such source, the SwissFEL, will be built at the Paul Scherrer Institute in Switzerland. To achieve optimal lasing conditions for the SwissFEL, the baseline design foresees to generate electron bunches with charges between 200 and 10 pC and lengths between 10 picoseconds (ps) and a few femtoseconds (fs). Accurate and non-destructive monitoring of such short bunches during the FEL operations is not easy to implement. One possible solution of this problem can be provided by Electro Optical (EO) methods [2].

EO electron bunch length measurements are based on the interaction between the electric field E_{THz} (which is in a THz range) generated by the electron bunch and the EO laser pulse in an electro optically active crystal (such as GaP or ZnTe). The electric field can be either the direct bunch Coulomb (near) field if the measurements are performed inside the beam pipe or coherently emitted synchrotron radiation outside of the beam pipe. The field induces a birefringence in the crystal that is proportional to the applied field. This changes the polarization state of the laser pulse, which can be detected by standard optical methods.

The simplest way to determine the E_{THz} shape is to sample it with much shorter laser pulses. If the laser pulses pass a crystal with some variable time delays Δt relative to the E_{THz} field, then they overlap with different parts of the E_{THz} field and experience different polarization rotations. By measuring the degree of the polarization rotation as a function of the time delay Δt, the E_{THz} can be mapped. It is clear that this method is not single shot and requires stable measurement conditions (such as electron bunch shape and EO laser intensity) over many pulses as well as a low relative arrival time jitter between EO laser pulses and E_{THz}.

Alternatively to the E_{THz} field sampling, the E_{THz} pulse can be overlapped with a longer EO laser pulse, which will transform the entire temporal structure of the electron distribution in a bunch on a single laser pulse. The EO laser pulse is stretched in a dispersive medium or by a grating compressor leading to a chirped pulse. A temporal profile of the E_{THz} field is linearly encoded into the laser pulse and can be determined by measuring its spectrum. This single shot technique is known as EO spectral decoding. It is not sensitive to the time jitter between EO laser and E_{THz} signals but requires a treatment of frequency mixing problems.

EO BUNCH LENGTH MONITORING SETUP

A prototype of a bunch length monitoring system (BLMS) for the SwissFEL was created at the PSI. It was successfully tested in the conditions of the Swiss Light Source (SLS), the only electron machine that was available at PSI before the SwissFEL Injector Test Facility was put in operations.

There are at least two places in the SLS ring, which can directly benefit from BLMS operations. The first one is the diagnostic section at the end of the injector where typical bunch lengths are from 2 to 20 ps. The second place is the SLS FEMTO facility [3]. The facility generates coherent fs X-rays based on the beam slicing technique, which utilizes the resonant energy exchange between a long (~35 ps) electron bunch and a fs laser pulse in a wiggler or an undulator. Beam slicing also results in a small (~100 fs) gap in the longitudinal distribution of the bunch that radiates coherently in a THz range. As the bunch continues to move around the ring, the gap quickly spreads and fills with electrons. Real time monitoring of this dynamics can be used to optimize the beam slicing quality.

Because both those places are important for the SLS, it was decided to create the first BLMS as a mobile beam diagnostics station, which could easily be moved around the SLS ring.

We note that most of BLMS parameters presented in this paper are specific for the SLS but can easily be adjusted to fit the SwissFEL case.

Main BLMS Components

The core of the BLMS system is an advanced Ytterbium fiber laser unit [4] together with pulse generating, mode locking and synchronization electronics.

The laser unit was designed and built in the frames of the collaboration between the PSI and the University of

#pavel.chevtsov@psi.ch

Bern. It consists of a 50 MHz oscillator and a 1 MHz amplifier. The oscillator cavity is split in free space and fiber sections. A piezo fiber stretcher can correct the resonator length on a µm scale in order to synchronize the repetition rate to the master clock. Cavity length adjustments on a larger scale are done by free space mirrors movable by remotely controlled stepper-motors. The required EO laser operational stability is achieved with the use of laser diode controllers (ITC5xx Series, Thorlabs) and a set of heating elements connected to temperature controlling devices (CT16A, Minco). The laser diode controllers have a GPIB interface to remote computers. Temperature controllers can talk to the external world via serial (RS-232) ports.

The BLMS electronics was developed and built at the PSI. It is based on the FPGA technology and has a modular design. All active elements form two compact 19″ electronics modules: the laser synchronization (LS) unit and the universal pulse divider and generator (UPDG) unit. Both units implement RS-232 interfaces to remote computers, which allows one to relatively easily integrate them into any control system.

BLMS Controls Software and Hardware

BLMS operations at the PSI are automated with the use of controls software based on EPICS [5]. The software consists of two main parts: the laser and the LS/UDPG control modules. Both parts utilize the EPICS Stream Device support [6] package to handle RS-232 and GPIB devices.

The controls software runs on three EPICS Input Output Controllers (IOCs).

The first one is a VME based single board computer MVME-5100 running VxWorks. This IOC handles

- a PSI timing event receiver (EVR, Micro-Research) board that is used to trigger electronics equipment based on any event available from the PSI timing event distribution system;
- BLMS stepper-motors on the basis of a MAXv-8000 (Pro-Dex) VME card and standard PSI motor drivers;
- serial BLMS control devices via RS-232 ports provided by TIP-866 IPAC modules (TEWS Elektronik).

Another IOC is a LINUX microIOC (Cosylab) handling GPIB control components via an Agilent E5810A LAN GPIB multiport controller. It also runs control applications automating major BLMS operations.

Finally, the EO data acquisition and control server is implemented as a softIOC embedded on a fast digital oscilloscope running WindowsXP.

All BLMS components, except the laser unit, are placed in a mobile 19″ electronics rack. The rack is also equipped by its own EO PC. This PC primarily acts as a boot computer for the VxWorks IOC. As a result, the IOC is booted on any PSI sub-network automatically, by turning on its power, without changing its boot parameters. The EO PC also acts as a port server to access the IOC VxWorks shell and as an additional computing power for EO related EPICS development activities.

The following paragraph explains how the BLMS electronics and controls software work by way of an example of their operations for the SLS beam slicing diagnostics.

BLMS at Work

As it was shown above, EO measurements rely on a temporal overlap between a laser pulse and the electric field generated by the electron bunch in an electro optically active crystal. This requires a very good synchronization to a reference signal (RF) and a low jitter of the laser signal. On the other hand, the absolute timing is also important and needs to be taken care of. If the overlap is found once, the absolute timing should be the same after any required laser resynchronization.

In this context, a problem arises if two signals of different repetition rates have to be locked, such as, for example, the EO laser at 50 MHz and the RF at 500 MHz. As it is more precise to compare these signals at higher frequencies (the same phase mismatch at 500 MHz leads to a ten times lower temporal offset than at 50 MHz), the comparison is done at 500 MHz, which means that the 10-th harmonic of the laser repetition rate is locked to the RF. As a result, the laser can be synchronized to the reference in ten different ways in terms of the absolute timing. The following shows how this problem is solved for BLMS. It also describes how the whole synchronization system works.

The next signals are relevant for the EO diagnostics:

- RF – 500 MHz;
- EO fiber laser oscillator – 50 MHz;
- SLS revolution clock – 1 MHz;
- FEMTO slicing trigger – 2 KHz;
- SLS linac trigger – 3 Hz.

The revolution clock is synchronous to the revolution of a particular bunch in the SLS storage ring and the FEMTO slicing trigger is used for measurements of the sliced bunches.

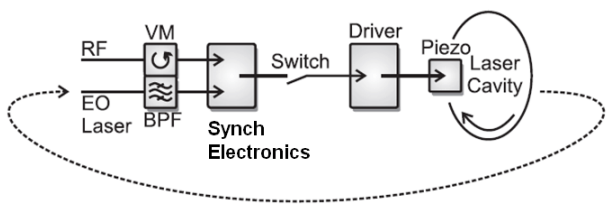


Figure 1: EO phase locked loop (PLL) scheme. The 10-th harmonic of the EO laser repetition rate generated by the bandpass filter (BPF) is locked to the RF, which can be shifted in time by the vector modulator (VM).

The 10-th harmonic of the EO laser repetition rate (see Fig. 1 above) is generated by a bandpass filter (BPF) and is compared to the RF, which can be shifted in time by a vector modulator (VM). The VM shifts the phase with a

resolution of 2^{12} steps per revolution (360°). At the RF repetition rate of 500 MHz this leads to a minimal step width of about 488 fs. Two signals are compared by the synchronization electronics, generating a correction signal. The switch between the electronics and the driver is remotely controlled in order to interrupt the PLL in case of any required interlock conditions or simply to switch the synchronization on or off. A piezoelectric device finally corrects the cavity length. The repetition rate of the laser serves as a feedback signal and is closing the loop.

In the second step, a reproducible starting point has to be found. This is done by the coincidence detector (see Fig. 2), which compares the repetition rates of the EO laser and revolution clock. As long as the signals are not synchronous the coincidence detector control software shifts the EO laser pulse in time by rotating the phase of the RF until the overlap is found. The accuracy of this method is in the order of ±5 ps, which allows one to find such an overlap very quickly.

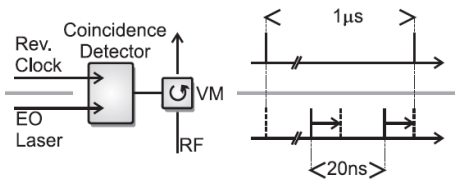


Figure 2: The second step of the EO synchronization. The coincidence detector shifts the EO laser pulse in time until coincidence to the revolution clock is achieved.

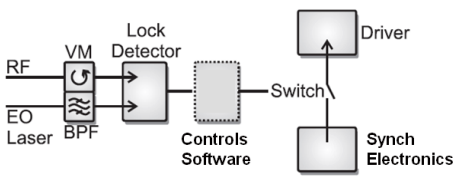


Figure 3: The lock detector control software interrupts the PLL if the offset frequency between the RF and the repetition rate of the EO laser becomes larger than 1 Hz.

In case of any malfunctions of the synchronization system, interlock signals have to be generated in order to interrupt the PLL and protect the piezoelectric fiber stretcher. It is done by the lock detector controls software (Fig. 3). As soon as the frequency difference between the RF and the 10-th harmonic of the EO laser repetition rate exceeds 1 Hz the output of the lock detector switches from logic high to low disconnecting the PLL link between the synchronization electronics and the piezo driver.

Between the oscillator and the amplifier the repetition rate is reduced to 1 MHz by an Acousto Optic Modulator (AOM). Because this pulse picker needs to be synchronous with the laser, it is triggered by the repetition rate of the EO laser. The BPF generates a sinusoidal signal at 200 MHz, which is connected to the clock of a resettable counter. The counter is reset by the 3 Hz SLS linac trigger.

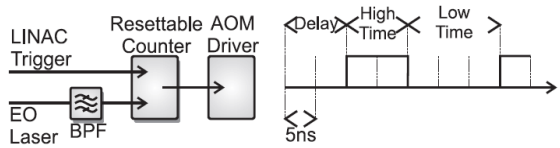


Figure 4: The trigger for the AOM is generated by a resettable counter. The delay, width and spacing can be adjusted in steps of 5 ns.

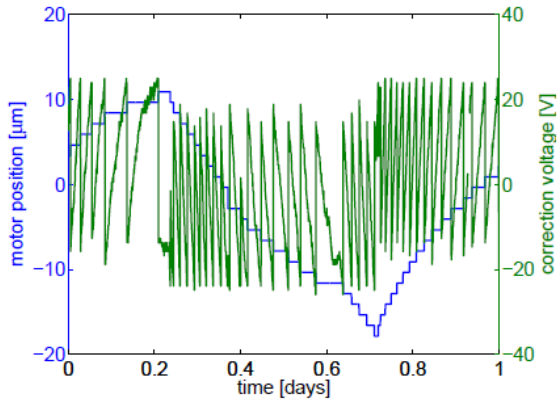


Figure 5: Typical automated corrections of the EO laser oscillator cavity length over one day. The green line is the piezo stretcher voltage, the blue line is the mirror position change.

The output is controlled in terms of a one-time delay, a high and a low time. As the device is counting with 200 MHz, these parameters can be adjusted in 5 nanosecond (ns) steps.

Besides handling the interlocks, the controls software ensures that if the piezo voltage exceeds a certain threshold then the required correction is done by one of free space mirrors. The Fig. 5 shows a typical behaviour of the piezoelectric stretcher (green line) and the mirror stepper motor (blue line), which is routinely provided by the controls software. We note that it is exactly what is expected of it.

The EO setup for the SLS beam slicing diagnostics [7] can be used for both techniques mentioned above: sampling and spectral decoding.

In case of sampling, the measurements of the signal amplitude as a function of the time delay are done by the oscilloscope software. The experiment is controlled by an EPICS high level application, which drives the VM, handles time delays, processes and stores the data.

Spectral decoding measurements are based on the use of a spectrometer. The spectrometer data are digitized and processed by embedded software, which makes these data immediately available for the control system.

CONCLUSION

The EO BLMS at the PSI has been in operations for more than one year. The system is fully integrated into the EPICS controls, which significantly simplifies all its functions. Bunch length measurement results obtained at the SLS show that the performance of the BLMS is absolutely adequate to its main task for the SwissFEL. Some additional work is required though, mostly for dealing with signal jitters in EO sampling procedures.

REFERENCES

[1] B. Patterson, R. Abela, H. Braun, et al. "Coherent Science at the SwissFEL X-ray Laser", New J. Phys. 12, 035012 (2010)

[2] E. Friedman and J. L. Miller, "Photonics Rules of Thumb: Optics, Electro-Optics, Fiber Optics, and Lasers", McGraw-Hill, 2004, 418 p.

[3] V. Schlott, et al. "THz Diagnostic for the Femtosecond Bunch Slicing Project at the Swiss Light Source", EPAC-2006, Edinburg, Scotland, 2006.

[4] F. Mueller et al. "A Compact Electro Optical Bunch Length Monitoring System – First Results at PSI", FEL-2010, Malmoe, Sweden, 2010.

[5] R. Lange, J. Anderson, et al. "EPICS: Recent Developments and Future Perspectives", ICALEPCS-2003, Gyeongju, Korea, 2003.

[6] D. Zimoch, et al. "Standardization of the DELTA Control System", ICALEPCS-1999, Trieste, Italy, 1999.

[7] F. Mueller, et al. "Electro Optical Sampling of Coherent Synchrotron Radiation for Picosecond Electron Bunches with Few pC Charge", BIW-2010, Santa Fe, NM, USA, 2010, p. 538-542.

LOW LEVEL RF CONTROL SYSTEM FOR CYCLOTRON 10MEV*

Jiang Huang#, Tongning Hu, Dong Li, Kaifeng Liu
Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

Abstract

The low level RF control system consists of a 101 MHz signal generator, three feedback loops, an interlock and a protection system. The stability of control system is one of the most important indicators in the cyclotron design, especially when the whole system has a high current. Due to the hugeness of the RF system and the complexity of control objects, the low level RF control system must combine the basic theory with the electronic circuit to optimize the whole system. The major obstacles in the research, which rarely exist in other control systems, lay in the coupling of beam and resonant cavity, requiring to be described by the transfer function between beam and cavity, the complex coupling between microwave devices and the interference signals of all loops. By introducing the three feedback loops (tuning loop, amplitude loop and phase loop) and test results from some parts of electric circuits, this paper unfolds the performance index and design of low level RF control system, which may contribute to the design of cyclotron with a high and reliable performance.

INTRODUCTION

The low level RF control system is an essential part of the RF system for CYCHU-10. The RF system should provide 12kW power to CYCHU-10 and the Frequency is 101MHz. The Specifications related to RF system are listed in Table 1.

Table 1: Main Specifications Related to RF System

Parameter	Value
Rated Power Output	12kW
Frequency Range	100.5~101.5MHz
Frequency Stability	20ppm
Output Impedance	50ohms

In the R&D of the CYCHU-10, a high power RF system has been established, which contains a klystron and its power supply and low level RF control system [1].

The low level RF control system consists of a 101MHz signal generator, three feedback loops, an interlock and a protection system. The design is basically similar to those of others electron storage rings in the world and is based on the use of conventional, stability and well-proven equipments. The scheme of low level control system is shown in Figure 1.

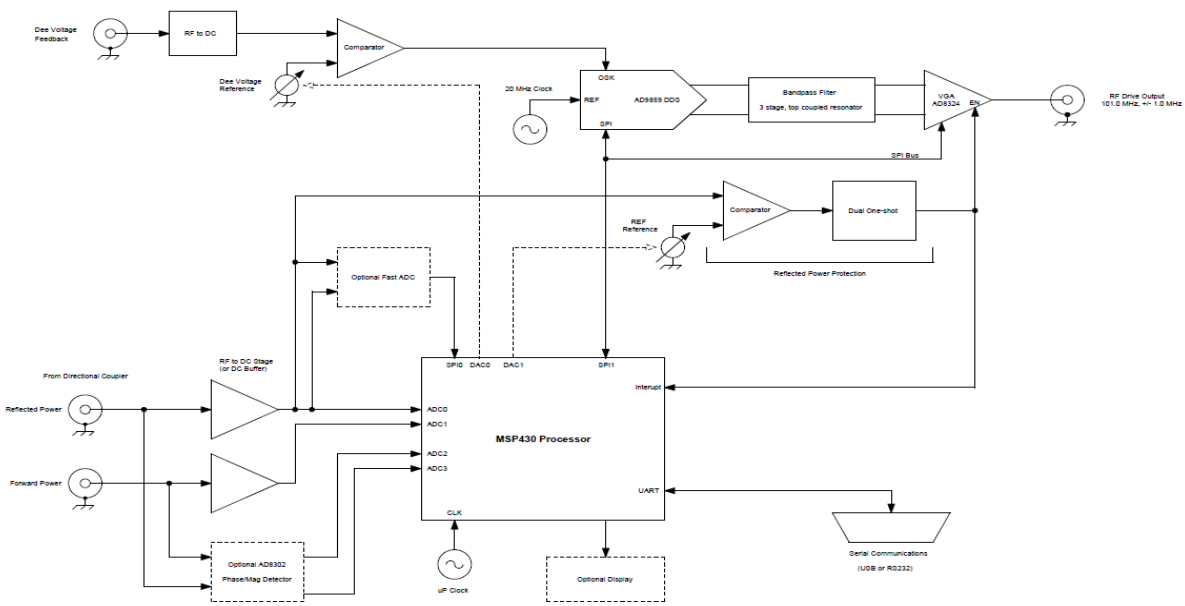


Figure 1:The scheme of low level control system.

*Work supported by National Science Foundation of China (No. 10435030)
#hjianghust@qq.com

FEEDBACK LOOPS

In the low level control system, the reference RF signal is generated by a DDS signal generator (AD9859). AD9859 is a great chip for synthesizing RF signal, which integrations a 10-bit DAC and provides 32-bit frequency tuning word. The chip also has great frequency resolution which can be 0.093Hz when system clock is 400MHz [2]. The output signal f_o is decided by system clock frequency f_s and frequency tuning word (FTW), the relationship is presented in the equation (1).

$$\begin{cases} f_o = (\mathrm{FTW})(f_s)/2^{32} \\ f_o = f_s \times (1-(\mathrm{FTW}/2^{32})) \end{cases} \quad (1)$$

In addition, the amplitude of the RF system can be achieved by adjusting the external connection resistor's value. This resistor value (R_{SET}) sets the internal DAC full-scale output current (I_{OUT}). The relationship is presented in equation (2).

$$R_{SET} = 39.19/I_{OUT} \quad (2)$$

In the above expression, the current is in A and the resistor is in Ω.

Tuning Loop

The tuning scheme is shown in the Figure 1. First, make the loop open and search the tuner position by changing the DDS's FTW. Secondly, make the loop close and compare the Reflect power and Forward power signal by phase/mag detector, then give the results to the controller. And at last, the controller set up DDS's FTW by the serial communication port, we use RS-485 in this system. The bandwidth of this loop is set up around 1 Hz. This tuning loop controls the DDS, which changes the frequency along with the cavity frequency [3].

Figure 2: The waveform of forward and reflect power.

In addition, a RF switch is set up to the end of drive amplifier, which has great effect to solve the multipactor effect. During frequency search the reflected power port can be monitored to watch the progression of reflected power level versus frequency. The waveform of forward and reflect power is shown in Figure 2. Each pulse is likely to have large spikes at turn-on and turn-off, that's because rapid change in the RF envelope spreads energy far away from the cyclotron resonance. As the frequency nears resonance and SWR becomes low enough, the controller shifts into Operate mode and applies drive continuously.

Amplitude Loop

RF signals picked up from the Dee Voltage pickup at the cyclotron, and compared with the Dee Voltage reference. Then the compared signal sent to the AD9859's OSK port, which can achieve close-cycle control of the amplitude. The protection reference value of the Dee Voltage and reflected power is setting up by the control program. Main Specifications of RF control system is shown in Table 2.

Table 2: Main Specifications of RF Control System

Parameter	Value
Power Requirement	+24VVDC, ~120mA
Forward Power Feedback Input	+10.8dBm (12 kW)
Reflected Power Feedback Input	+10.8dBm (12 kW)
Dee Voltage Feedback Input	+17dBm maximum
Forward Power Monitor Output	0-1.85VDC for 0-12kW
Reflected Power Monitor Output	0-1.85VDC for 0-12kW
Dee Voltage Monitor Output	0-2.5VDC for 0-40kV

Phase Loop

This loop is to keep phase of the fields in the cavity locked with the signal generator. RF signals picked from the direct coupler which in the front of the cavity, then sent to MSP430 processer to control DDS. The phase loop will also compensate the phase change with the RF power variance, due to the power amplifier, the circulator, the klystron, the driving electronics and so on. The components of the driving electronics are designed to have a small phase variation over a wide operating range.

The Phase Detector (PD) is the key component. The PD is a device with rather constant sensitivity against large power variations. This will prevent the effect of amplitude modulations on the operation of the loop.

SYSTEM TEST

The low level RF control system consists of a 101MHz signal generator, three feedback loops, an interlock and a protection system. It connects with the cyclotron control system by serial communication port (RS485). The RF control system is shown in Figure 3.

Monitor ports for forward power, reflected power and Dee voltage RF feedbacks are available on the front panel. These outputs, from the respective RF-to-DC converter stages, are unity-gain buffered and can drive an

oscilloscope or other >1kΩ load. No filtering is applied to the monitor outputs so response time is equal to the RF detector response. The level at the forward and reflected ports corresponding to 12 kW is approximately 1.85 VDC. The picture of control system is shown in Figure 3.

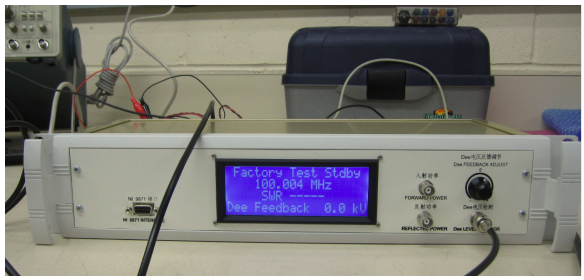


Figure 3: The picture of control system.

To prove the function of the control system, we do bench-top testing. It is possible to simulate the cyclotron load and demonstrate the controller operation using a resonant cavity. The Bird Electronics Model 211-29-05 shown Figure 4 can be adjusted to give the desired SWR, resonant frequency and feedback port attenuation for bench-top testing.

Figure 4: The picture of resonant cavity.

For bench testing an outboard amplifier is used to boost signal levels high enough for sampling by a directional coupler. The controller output drives the input of the amplifier and the controller forward and reflected inputs receive feedback from directional coupler. The picture of amplifier and directional coupler is shown in Figure 5.

Figure 5: The picture of amplifier and directional coupler.

A LabView utility program is used to set up and operate the controller. This program has a Monitor tab, shown in Figure 6, with easy-to-use command inputs and feedback of important parameters.

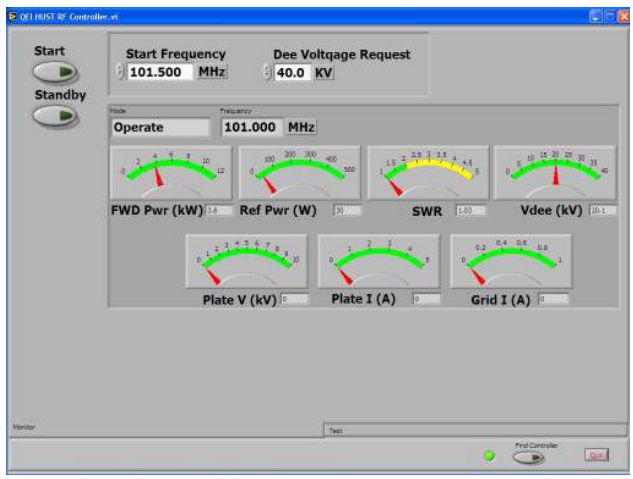


Figure 6: LabView Control Program – Monitor Tab.

The scope should show strings of four tuning pulses for each frequency step in the search. As the frequency approaches 101.00 MHz the height of the reflected pulses will become smaller and the SWR value will eventually fall below the Operate Mode threshold. At this point the controller output switches to steady-state. The process of searching resonance frequency is shown in Figure 7.

Figure 7: The process of searching resonance frequency.

CONCLUSIONS

The low level RF control system for CYCHU-10 has feedback loops which monitor Dee Voltage, Forward Power and Reflected Power. This allows the system to seek and follow the resonant frequency of the cyclotron cavity and maintain a constant Dee Voltage. The controller operating parameters are communicated via an RS485 serial interface. The system communicates using a message packet that carries commands to the system and feedback from the controller regarding its status.

The bench-top testing results of control system show that the system can achieve the designed functions. The further debugging with the whole RF system for CYCHU-10 will be carried out.

REFERENCES

[1] In Su Jung and Dong Hyun An et al, “Design of KIRAMS-13 RF system for regional cyclotron center”, Proceedings of the 17th International Conference on Cyclotron and Their Applications, 2007, p. 353-355.

[2] Analog Devices Inc. 400MSPS, 10-bit, 1.8V CMOS Direct Digital Synthesizer AD9859 EB/OL. http://www.analog.com

[3] Cai Zhiyuan, Wei Wei, Ma shaohus. Design of Program-Controlled Current Source Based on DDS Technology. ICEEE 2010, 2010

HIGH-SPEED DATA HANDLING USING REFLECTIVE MEMORY THREAD FOR TOKAMAK PLASMA CONTROL

S.Y. Park, S.H. Hahn, and W.C. Kim, NFRI 113 Gwahangno,
Yuseong-gu, Daejeon, 305-333, Korea
B.G. Penaflor, R.D. Johnson, D.A. Piglowski, M.L. Walker,
General Atomics, P.O. Box 85608, San Diego, CA 92186-5608, USA

Abstract

The Korea Superconducting Tokamak Advanced Research (KSTAR)[1] plasma control system (PCS) is defined as a system consisting of electronic devices and control software, which identifies and diagnoses various plasma parameters and calculates appropriate control signals to each actuator to keep the plasma sustained in the KSTAR operation regime. The KSTAR PCS consists of a linux system with 8 processors and both analog and digital data acquisition methods are adapted for fast real-time data acquisition up to 20 kHz. The digital interface uses a reflective memory (RFM) technology to share data among various subsystems of KSTAR. RFM technology has been adopted as the real time communication method to enable PCS to interface with the actuators and to do interprocessor communications inside the cluster. To handle the fast control of the RFM data transfer, the communication using the RFM with the actuators and diagnostics system is implemented as a thread which is assigned to a separate process.

INTRODUCTION TO THE PLASMA CONTROL SYSTEM

The plasma control system is composed of real-time computers for feedback calculations, a diagnostic system for plasma information, and a communications interface with actuators. The PCS acquires plasma data from the diagnostic system and performs a feedback control loop to obtain plasma properties. Figure 1 shows the plasma control system structure. The PCS feedback algorithm calculates the difference between target and measured values, and decides how much coil current is needed in order to reduce the error from the target. Next the PCS sends coil voltages to the magnet power supply (MPS) for the desired poloidal magnetic field (PF) and receives coil current measurements from the MPS. In KSTAR this feedback operation is performed over an optical network consisting of reflective memories [2] which are high-speed replicated shared memories with up to 256 nodes featuring very low latency and wide throughput. Although the performance of the initial system [3] with a single process was acceptable, performance demand for a shaped plasma required faster control cycles up to 50 us, as well as increased interprocess communications for sophisticated magnetic controls. In 2011, the amount of I/O data exchanged in a single cycle was about 1 kB, hence the old PCI-based method was not suitable for the requirement because of 3 us time overhead for each access of the RFM space.

A method utilizing the dedicated thread for RFM is introduced for the following purpose: 1) to minimize

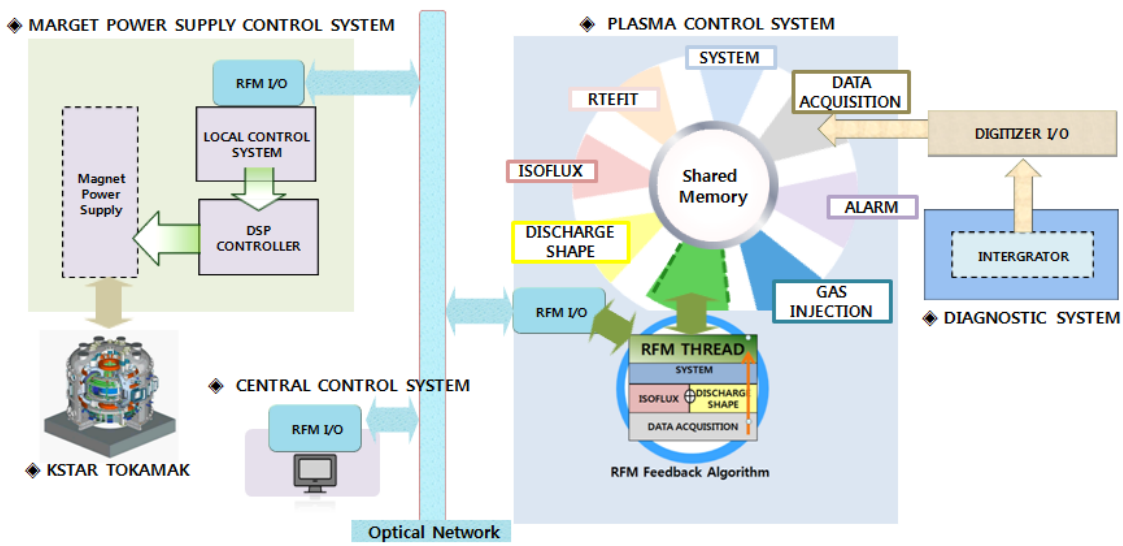


Figure 1: The structure of the plasma control system.

intrinsic time overheads for the RFM access, 2) to transfer all the RFM data within 50 us, and 3) to gather all the RFM data generated in other 3 cpu's running in the PCS via shared spaces. In this paper, we describe the principles of the RFM thread and the feedback algorithm for the PF coil control.

OPERATION OF THE RFM THREAD

The structure of the RFM thread is shown in Fig. 2. The RFM thread is spawned as a child process of one of real time processes and assigned to a separate physical processor at the start of each shot and killed at the end of the shot. The main operation of the RFM thread is to copy the RFM data to a preassigned shared memory area at every cycle to use in the feedback algorithm. In the KSTAR PCS, this shared memory is shared by all the real-time processes and is referred to as the real-time heap memory (rtheap) [4]. The rtheap is a shared memory that contains various constants and pointers to all of the other structures in the memory of the real time process. The RFM thread synchronizes itself with the real time control cycle by accessing the rtheap time counter delivered from the digitizer.

When the RFM thread starts, it calls the initialization function that creates the RFM handle, and allocates the internal memory used for communication with the RFM. If the initialization routine is successful, this routine returns the virtual address of the DMA memory address. After performing the initialization routine the RFM thread executes a continuous loop. At first the RFM thread reads values from a fixed memory area which is assigned for real-time measurement values for PF coil currents and voltages, and flags indicating the software limit defined by the central control system (CCS) or the hardware limit defined by the MPS. These measurement values are moved to an "internal buffer" memory area (see Fig. 2) and are saved in the shared memory in the same format as the acquired DMA data from the digitizers. The feedback control algorithm gets the data from the shared memory after one cycle. After reading the data, the thread waits for the current time to change by monitoring the fixed area of the rtheap. For the time synchronizations of the main and the RFM thread, a counter is used for the "new time". This counter synchronizes the internal CPU clock count of the cpu1 real-time process to the clock counter provided by the external clock of the digitizers [5]. The RFM loop waits for the counter change in order to catch this new time when the next cycle starts to determine the feedback command.

The process of determining feedback command of the next cycle is performed by the PCS feedback algorithm in the spawning process. As shown in Fig. 2, when deciding the PF coil command of the current cycle, the measured coil current of the preceding cycle is used; this coil current is the value saved in the shared memory area before the cycle. The reason for using measured value at the preceding cycle for obtaining the command of the current cycle is to reduce the feedback error occurred at the preceding cycle. The PCS feedback algorithm acquires the error value of the previous cycle by subtracting measured current in the previous cycle from the target current, and calls the PID function for adding compensation voltage at the next cycle. The answer from the PID function is sent to the RFM thread as the next PCS command. The RFM thread gathers all the commands for each MPS and writes the command to the internal buffer in order to do a DMA transfer to the fixed RFM area assigned for the MPS.

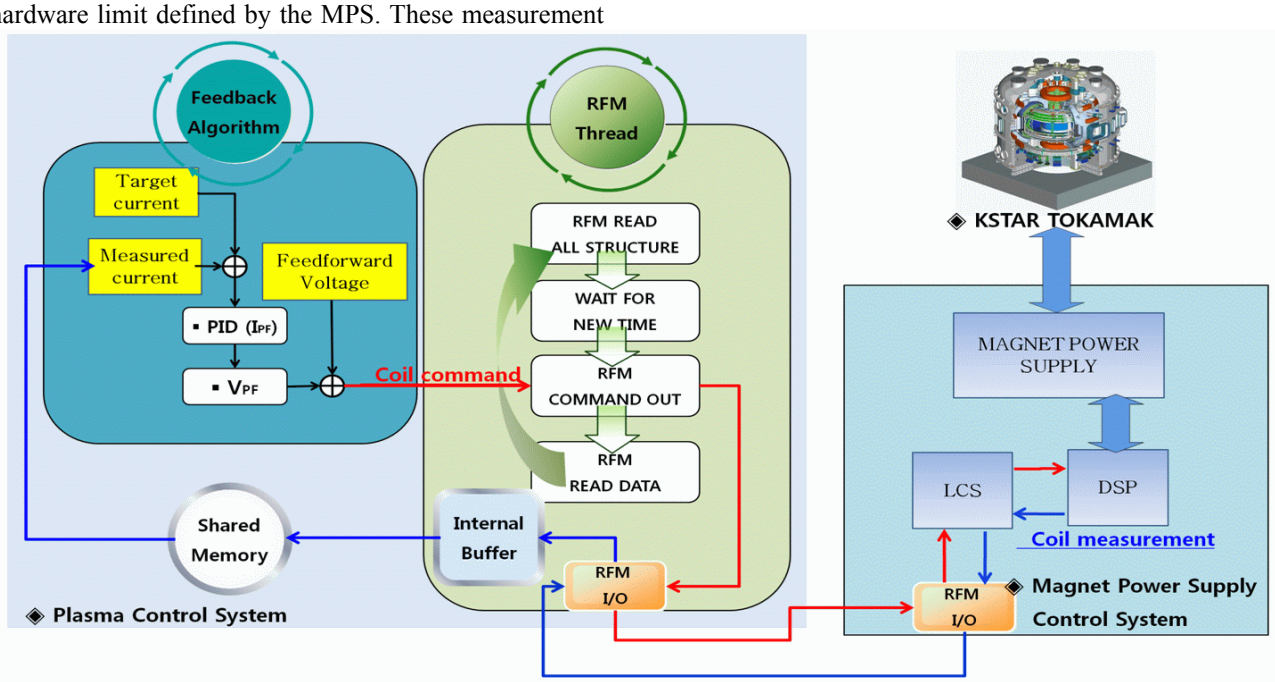


Figure 2: The RFM thread and PCS feedback algorithm.

Timeout Handling of the RFM Thread

The RFM thread has a few event handling algorithms in order to monitor its execution cycle, sync with the main thread and to prevent itself from endangering actuators under control of the PCS. The self-monitoring of the RFM loop is done by a time-out count. If the current time has not changed during the time-out period or the real-time process exits from the real time mode for any reason, the RFM thread fills zeroes in the PCS command structure which it sends to the MPS. Since the MPS (and its DSP controller) accepts PCS commands only during real-time mode, this will cause the MPS to exit out of PCS control and have the DSP controller take over control of the PF power supply. Hence the PF power supply can avoid a dangerous situation when the PCS is out of control by its own internal delay. This time-out period is set to 5 ms, which corresponds to the effective response time of a single MPS voltage command [3]. During this time-out period, the RFM thread also checks whether the main process is not updating the new time counter. If it is not, the RFM thread informs this situation to the central control system by updating a fault code so that it can spread termination signals to the other systems to abort the discharge.

Another example is a "software watchdog". This is a counter which is increased by 1 on each control cycle. Since the counter is shared as a "timestamp" in the RFM area, this counter can be monitored by any device under the same RFM network as PCS. As a kind of heartbeat, the watchdog counter is monitored by each MPS and the CCS so that it can check communication shutdown on the RFM. If the value exceeds the limit, this value sets to zero again.

When the real time mode is ON by the PCS, the CCS monitors the software watchdog counter and checks the value every 10 ms. The CCS makes a fault code to the other system, if this value is not changed by 10 ms -- which could imply that the PCS has either a communication fault or a serious internal fault such as power down or real-time process hang-up.

Some Issues about the RFM Thread

- The time counter used in the RFM operation is acquired from the digitizer. Hence the clock source of the digitizer should be accurate. If the clock source does not have good accuracy, the RFM thread will be terminated because of a timing mismatch problem.
- Although the data processing speed of the RFM thread is fast, if the MPS interface system does not operate as we expected, there is a possibility that the RFM thread receives the same data during a couple of cycles. This could cause a saturation command issue in the P loop; however, in a practical manner, the design of the algorithm considering the delays by the slower update of the measurement can avoid the saturation issue.
- When writing to the PCS command to the RFM area, the RFM thread assumes that the values in the memory structure are the most recent written by the real time process. The RFM does not keep the previous value before it is updated by the new data. Due to this property, the RFM data written by two different devices is not exactly synchronized in time. Nevertheless, the assumption is true in most cases and the data is acceptable as the most recent one if the read cycle is faster than each writing cycle by the actuators, which fits to our case.
- If the RFM thread doesn't synchronize with the real time process because of some problem such as accessing wrong memory area or time delay for writing to memory, the thread could miss cycles. Several missing cycles are reported when the RFM initialization function is called. This is due to the time overhead of the initializing hardware handle. In order to avoid timeout errors, the real time mode flag is updated as ENABLED after those missing cycles are gone, so that the PCS can get synchronized data from the MPS and the MPS can receive correct feedbacks.

The Command Structure Sent to the MPS

The entire PCS command structure is read by the local control system (LCS) of each MPS and sent to the corresponding DSP, which actually communicates with the power supply (see Fig. 2). The DSP reads each field of this command structure and decides the way of the coil control. Table 1 shows the PCS command structure sent to the MPS.

- "control method" indicates how the PF command should be interpreted (voltage or current)
- "current direction" indicates the charging direction of the PF coil. (forward or reversed)
- "timestamp" is used as a software watchdog in each control cycle (increased by one on each cycle)
- "real time mode " tells the MPS when the PCS starts the real-time feedback cycle for them.

The Data Structure of the MPS and CCS

The amount of data that is received from a single power supply is 52 bytes, but the total amount of data is about 1 kB since there are several power supplies that should be controlled by the PCS. Table 2 shows the data structure of the MPS. The RFM thread receives all of the data that is sent by each power supply at once using DMA transfer. This is possible since the memory area corresponding to each power supply is arranged sequentially.

Table 3 shows the structure used for the CCS. There is a field that indicates current time of the PCS. This field is just for reference; the CCS operates in real time mode and has the PCS fault code which is set by PCS when unusual situation is occurred. When the CCS detects the PCS fault, the CCS informs this fault code to the other systems.

Table 1: Command Structure Sent to MPS

Parameter	Description
Id	Magic id
Control method	Indicates command type of the PF coil 0: current command 1: voltage command
Current direction	Initial charging current direction 0: forward 1: reversed
Time stamp	RFM watch dog counter
rt_mode	Indicates the PCS is in real-time mode
PF command	PCS command to send to MPS
Test variable	Dummy field for test
Current trajectory	Current trajectory when voltage is selected

Table 2: Data Measurement Structure from the MPS

Parameter	Description
Id	Magic id
Voltage	total voltage form MPS
Current	total current form MPS
Time stamp	RFM watch dog counter
D-axis current	D converter output current
Y-axis current	Y converter output current
Flag	over current flag
D-axis alpha	D converter alpha degree
Y-axis alpha	Y converter alpha degree
QP voltage	Quench protector voltage
Bris voltage	Bris voltage
D-axis voltage	D converter output voltage
Y-axis voltage	Y converter output voltage

Table 3: Data Structure for the CCS

Parameter	Description
Id	Magic id
PCS current time	PCS current time
PCS fault code	Indicates what kind of fault is occurred in the PCS
Force PCS abort	Central controller uses this value to abort the PCS

CONCLUSION

We were able to transfer all RFM data within 50 us, and control the PF coil efficiently using the RFM thread. We could increase the RFM memory read/write performance by rearranging the RFM memory sequentially and using the DMA transfer. We could identify that we can manage the RFM memory efficiency by using the RFM thread.

There are some sensitive issues. We have a plan to increase the amount of data through the RFM in the future; hence, we need to increase the RFM memory performance and also upgrade the interface devices for improved communication with the PCS in the RFM network and also for better coil control.

Currently we plan to improve performance by upgrading the RFM memory card to the PCI express bus format and to improve the algorithms to safely handle other possible unintended situations.

REFERENCES

[1] G.S. Lee, J.Y. Kim, S. Hwang, C.S. Chang, H.-Y. Chang, M.H. Cho, et al, Nuclear Fusion, 40 (2000) 575.
[2] http://www.ge-ip.com/
[3] S.-hee Hahn, M.L. Walker, K. Kim, H.S. Ahn, B.G. Penaflor, D. a Piglowski, et al., Fusion Engineering and Design, 84 (2009) 867-874.
[4] B.G Penaflor, J.R Ferron, M.L Walker et al, SOFT, 16-20 Sep 1996, Lisbon (Portugal)
[5] S. H. Seo et al, The 6th IAEA Technical Meeting Control, data Acquisition, and Remote Participation or Fusion Research 4-8, June, 2007, Inuyama, Japan

BPM SYSTEM AND ORBIT FEEDBACK SYSTEM DESIGN FOR THE TAIWAN PHOTON SOURCE

C.H. Kuo, Jenny Chen, Y.S. Cheng, P.C. Chiu, K.H. Hu, K.T. Hsu, C.Y. Wu,
NSRRC, Hsinchu 30076, Taiwan

Abstract

Taiwan Photon Source (TPS) is a 3 GeV synchrotron light source which is being in construction at NSRRC. The BPM electronics with the latest generation FPGA and new mechanical form factor to enhance functionality of current generation products will be employed for the TPS. The prototype BPM electronics is testing. To achieve the stringent orbit stability goal of the TPS, orbit feedback system is designed to eliminate beam motions due to various perturbation sources. The design and implementation plan of the BPM system and the orbit feedback system are summarized in this report.

INTRODUCTION

The TPS [1] is a latest generation of high brightness synchrotron light source which has been under construction at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan since 2010. It consists of a 150 MeV electron Linac, a 3 GeV booster synchrotron, and a 3 GeV storage ring. The design of the storage ring has 24 cells. There are 7 BPMs and 7 horizontal/vertical correctors are winding on the sexupoles in each cell. Current generation state of the art BPM electronics is designed and delivered more than 5 years ago. To avoid obsolesce of the components, to take advantages of advanced devices and enhance functionality, it was decided to adopt new design of BPM electronics. To satisfy stringent orbit stability requirement of the TPS, low noise corrector power supply system, and orbit feedback system are also designed.

STORAGE RING BPM AND CORRECTOR LAYOUT

The design of the storage ring has 24 cells, each cell equipped with 7 BPMs and 7 horizontal/vertical correctors are winding on the sexupoles in each cell. The lattice layout is as Fig.1. The locations of BPM and correctors are to satisfy DC orbit correction under the constraint of installation space. Since vertical beam size at centre of straight section for insertion devices is around 5 μm in sigma, better than 0.5 μm beam stability are required. To meet the tight orbit stability requirement, an orbit feedback system is indispensable. Standard correctors are side winding on the sextupole magnets. These kinds of correctors could provide about 500 μrad kick while their bandwidth could be limited only several tens of Hertz due to the eddy effect of the alumina vacuum chamber. The vacuum chamber is 4 mm thick elliptical chamber with 30 mm and 60 mm in minor axis and major axis respectively. Due to eddy current effect, the possible bandwidth in horizontal and vertical plan is around several tens Hertz. This bandwidth is not sufficient to eliminate perturbation below 100~200 Hz. Therefore, extra four horizontal/vertical correctors per cell will be installed on the bellows site as shown in Fig. 1 to obtain higher correction bandwidth. These correctors have fast response but smaller kick strength around 30 μrad. Thus the orbit feedback system plans to use two kinds of correctors simultaneously. The DC component of the fast correctors will transfer from fast to slow correctors smoothly and avoid saturation of the fast correctors as well as provide capability to suppress orbit drift.

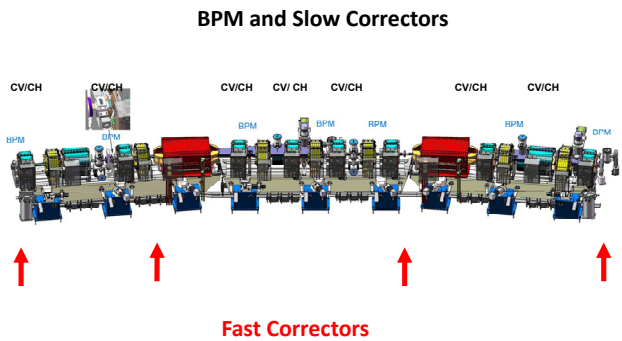


Figure 1: The slow (winding on sextupoles) and fast correctors layout in one cell of TPS storage ring.

BPM SYSTEM

BPM Electronics

The TPS BPM electronics was contracted to the Instrumentation Technologies in May 2011 for the Libera Brilliance+ as BPM electronics which is the latest member of the Libera family. The family covers the requirements of wide variety of the circular light source machines [1]. The BPM electronics delivers unprecedented possibilities for either building powerful single station solutions or architecting complex feedback systems. New electronics allows even more extensive machine physics studies to be conducted due to large data buffers and the new true turn-by-turn position calculation. There are 168 BPM units at the storage ring except some special BPM at straight sections. All 168 BPM electronic modules will installed at 48 BPM platforms. The first field tests of the new product had been performed on real beam at Taiwan Light Source (TLS) [2, 3] to test validation for the usage for the TPS. Figure 2 shows the tested installation.

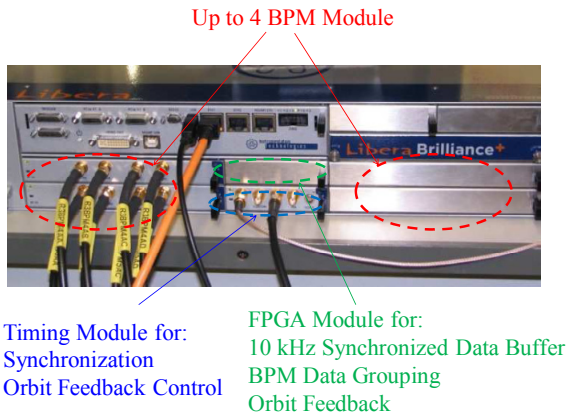


Figure 2: Libera Brilliance+ installation in the TLS.

Intensive test for performance and functionality were performed for the prototype to ensure that the new electronics satisfy the requirement of TPS. The test combined by using simulated signals from signal generator as well as real beam of the existed 1.5 GeV synchrotron light source. The long-term stability has been tested for bench measurement where input power level was set to several power level between -20 dBm to 0 dBm. Position stability is around 100 nm in rms for both planes for one day record. The temperature dependence has also been tested and almost could be ignored while ambient temperature was controlled within ± 0.1˚C. The resolution of FA data at 10 kHz meet specification where σ x and σ y is less than 150 nm. Besides, turn-by-turn data has been achieved 10 μm resolution for short bunch train.

The real beam tests at the TLS have also been performed. SA and FA reading had been acquired and analyzed for the stored beam. Current dependency affects position variation could be within 3 um from 30 mA to 360 mA. The instrument also possesses a useful feature which provides two approaches to process from ADC data to turn-by-turn data. One is classic DDC approach; another is time domain processing (TDP). This functionality is useful for peculiar filling pattern.

The preliminary testing ensures that the Libera Briliance+ BPM can meet the requirement of TPS storage ring. To simply the complexity, the same platform will adopt for the booster synchrotron also. There are 60 BPM units at the booster synchrotron. All 60 BPM electronic modules will be installed at 24 BPM platforms around the 24 CIAs.

BPM Data Access and Grouping

There are several data format flows are provided by the BPM platforms with EPICS interface, include of 10 Hz rate data for DC closed orbit correction, turn-by-turn data with software/hardware trigger and on demand access for accelerator physics study, streaming 10 kHz fast data for orbit feedback application. The fast data is also very useful for beam diagnostic.

The BPM platform is composed of a COM Express CPU module running Linux to control up to 4 BPM modules. Embedded EPICS server on this module could deliver 10 Hz rate access of BPM information, includes matrix elements of the orbit feedback system as well as handle the basic configuration.

A GDX FPGA module will be installed at the BPM platform to support 10 kHz synchronized data buffer for 4 BPM modules, and group with the other BPM platforms for the whole storage ring together. The fast orbit feedback control and fast corrector interface protocol are also processed in this module. There are four SFP ports on the front plane. There are several LVDS links in the BPM modules that can collect all ring BPM data with 10 kHz rate. The data is through the bi-directional multi-gigabit links to the adjacent nodes. Each FPGA module will serve as one node of the BPM grouping around the ring. We refer to this functionality as BPM grouping. The communication of the grouping is done by two SFP ports. The bitrate of the grouping can be up to 6.25 Gbit/sec. One of the remained SFP port will be used as fast corrector control interface with AURORA protocol. The last SFP port will provide functionality for grouping data output either use AURORA protocol or UDP/IP protocol, to support the slow corrector control or the diagnostic related application. The Ethernet packet supports jumbo frame to satisfy all BPM data grouping. Due to BPM data grouping and orbit feedback control use the same FPGA module, orbit feedback is also part of BPM platform.

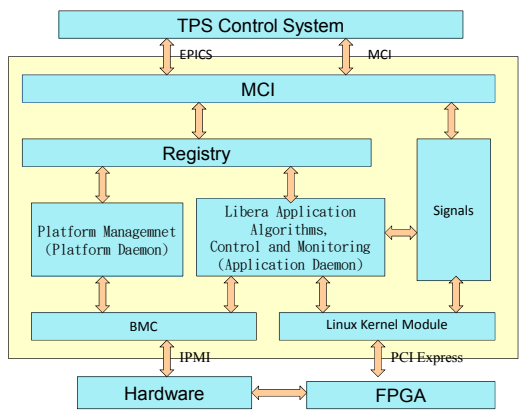


Figure 3: BPM platform software structure.

BPM Platform Software Structure

BPM platform software is organized in a few layers, see fig 3. In the bottom layer there are BMC library and Linux kernel module which communicate with FPGA and hardware through IPMI (Intelligent Platform Management Interface) and PCI Express respectively. On the top of that there is signal processing library and the application daemon for control and monitor algorithms processing. Beside application there is also platform management (Platform Daemon), which is responsible for monitoring health status of the hardware. The application and

platform daemon communicate with outer world through the MCI (measurement and control interface) which uses registry and signal library underneath for accessing application properties, signals, changing the application behavior, etc. On the top of MCI various adaptors to different control systems can be implemented [4].

ORBIT FEEDBACK SCHEME

The TPS aluminium vacuum chamber of storage ring is 4 mm thickness that prevents fast orbit correction by standard correctors in both planes which are back winding on 168 sextupole magnets. In order to satisfy fast orbit feedback request, 96 sets of fast corrector at bellows site are added. So, the TPS orbit feedback system is a global orbit feedback system combined fast and slow corrector in one system.

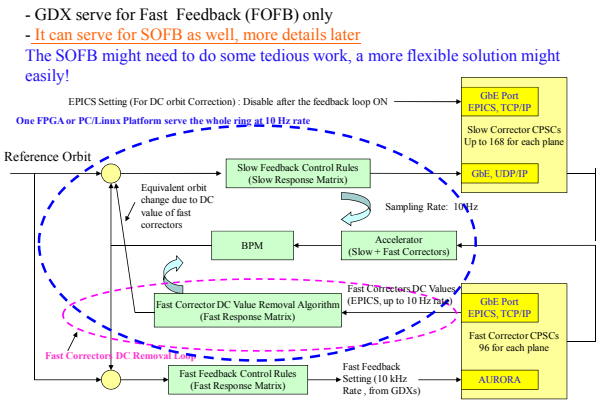


Figure 4: Control infrastructure in one cell of TPS storage ring power supplies.

Orbit Feedback Infrastructure

The infrastructure configuration of orbit feedback with fast and slow corrector is shown in Fig. 4. There are 24 cells in the TPS storage ring. Each cell will be equipped with two BPM platforms which can accommodate 7 BPM electronics at Phase I. Each BPM platforms has a slot which can allocate an FPGA module used for communication with the other module in other cells as a clockwise and counter-clockwise multi-gigabit redundancy link to group BPM data for the whole ring. Two FPGA modules of each cell will handle fast orbit feedback control algorithm and control 4 fast correctors in horizontal and vertical plane respectively. Correctors are controlled by a custom designed Corrector Power Supply Controller (CPSC) of each cell [5]. The CPSC module will be installed at slot of power supply crate to control up to 8 modules in the same chassis. At each cell, there are three CPSC modules installed at three power supply crates include of horizontal slow correctors (7 PS modules), vertical slow correctors (7 PS modules), and combined horizontal (4) and vertical (4) fast corrector. Fig. 4 shows

Besides fast correctors computation executed in FPGA modules, the slow corrector control will be handled by general purpose CPU. It would acquire the BPM grouping data through the rest SFP port of FPGA module configured as Gigabit Ethernet. The slow corrector will be controlled with 100 Hz update rate and be delivered at phase I to keep from fast corrector saturation. The slow corrector setting rate is expedited that is possible in the future. The possible candidate platform would be aTCA.

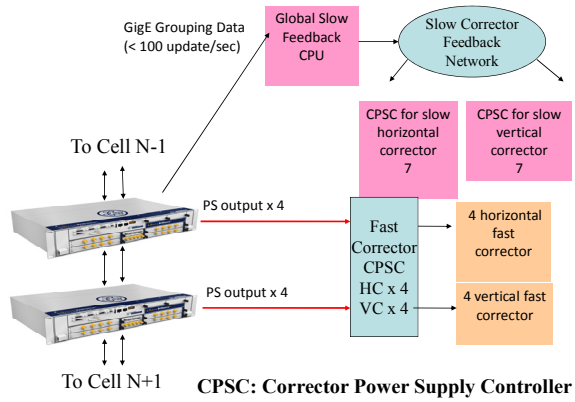


Figure 5: Configuration of global orbit feedback for corrector power supply.

Corrector Power Supply and its Control Interface

Fig. 5 shows the small power supplies applied for both slow and fast correctors and skew quadrupoles. There will be 8 power supply modules in one crate and the most left slot will be plugged in one CPSC. Besides general control, monitor and configuration, the fast correctors will be also applied for the fast orbit feedback. The CPSC with EPICS IOC is therefore dedicatedly designed and the embedded FPGA will handle fast update application. Built-in waveform and synchronization mechanism are also supported.

The power supply for both fast and slow corrector magnets in the range of ±10 Amp will be controlled by analogue interface directly. The CPSC is dedicated to be designed for both EPICS control system and fast orbit feedback application which is based on AURORA of Xilinx to support high speed communication and control. Fig. 6 shows the corrector power supply crate which includes 8 power supply modules and one CPSC at the most left slot. The CPSC with EPICS IOC is dedicatedly designed and the embedded FPGA will handle fast update application. Synchronization mechanism and built-in waveform are also supported. The setting reference of the corrector power supply is generated by 20 bits DAC and readback is from 24 bits ADC. This controller is also a embedded EPICS IOC for slow control. Fast setting ports are to receive fast data from orbit feedback module (GDX).

Control Rules Scheme

The response matrix R_s and R_f, which relates the orbit shifts to the slow and fast correctors respectively could be decomposed by singular value decomposition (SVD) as Eq. 1 and Eq. 2.

$$R_s^{+} = V_s \Sigma_s^{+} U_s^{T} \quad (1)$$

$$R_f^{+} = V_f \Sigma_f^{+} U_f^{T} \quad (2)$$

where R_s is 168×168 matrix; R_f is 168×96 matrix. These two loops of correction are operated at different platform but shared the same BPM data streaming. As well as the other light sources, to avoid saturation of fast correctors and counteraction of fast and slow loop, the different controllers would be applied for two loops respectively to separate the working frequency domain.

Figure 6: Corrector power supply crate. There are 8 power supply regulation modules in one crate and the most left slot will be plugged in one CPSC.

Fast Orbit Feedback Loop

The fast orbit feedback loop will be performed in the GDX FPGA modules in the distributed manner; each FPGA module control four fast correctors. The matrix elements download and feedback parameters will be done by the EPICS IOC located at the platform of the GDX FPGA module. All feedback loop of platform will be synchronous by grouping and external signal, such as feedback on/off.

Slow Orbit Feedback Loop

The slow orbit feedback loop processing will use PC running Linux. The individual BPM and fast corrector setting value in each cell will be combined in the local net, and transfer to a waveform type of EPICS interface to satisfy software real-time access request for slow orbit feedback control. The CPSC is also EPICS IOC that will receive slow setting value from typical EPICS interface or UDP packet port. Each CPSC module would extract their setting data by specific address within the packet.

Communication between Fast and Slow Orbit Feedback Loop

The plan FOFB system will working down to DC. A slow system will have to read the DC part of the current in the fast corrector from the CPSC module and transfer it on the slow correctors by multiply inverse of the slow transfer matrix. The two systems would not be independent; a reliable communication needs to be implemented between them. The main advantage of this solution is to reduce beam movements in the whole frequency range. This scheme were high successful in ALS and Synchrotron Soleil [6, 7].

SUMMARY

The design of the BPM system and corrector power supply system were performed for the TPS storage ring. The BPM electronics contract to the vendor in June 2011. All BPM electronics will be delivered in late 2012. Orbit feedback FPGA module is in call for tender process. Corrector power supplies will be delivered in later 2012 also. System integration is scheduled in 2013. Full scale test of the system will be possible in late 2013. Test with beam will be possible in 2014. Preliminary test for the system will be possible in 2013. All necessary hardware and software are in preparation.

REFERENCES

[1] http://www.nsrrc.org.tw/english/tps.aspx.
[2] P. Leban, et al., “First Measurements of a New Beam Position Processor on Real Beam at Taiwan Light Source”, Proceedings of the PAC 2011, New York, U.S., 2011.
[3] C. H. Kuo, et al., “Preliminary BPM Electronic Testing for the Taiwan Photon Source Project “, Proceedings of the IPAC 2011, San Sebastian, Spain, 2011.
[4] http://www.i-tech.si.
[5] http://www.D-Tacq.com.
[6] C. Steier, et al., “Operational Experience Integrating Slow and Fast Orbit Feedbacks at the ALS”, Proceedings of EPAC 2004, Lucerne, Switzerland.
[7] N. Hubert, et al., “Commissioning of Soleil Fast Orbit Feedback System”, Proceedings of EPAC08, Genoa, Italy.

AN OVERVIEW OF THE ACTIVE OPTICS CONTROL STRATEGY FOR THE THIRTY METER TELESCOPE

Mark J. Sirota, George Z. Angeli, Douglas G. MacMynowski, TMT Observatory Corporation, Pasadena, CA. 91105, U.S.A.
Terry Mast, Jerry Nelson, University of California, Santa Cruz, CA. 95064, U.S.A.
Gary Chanan, University of California, Irvine, 92697, U.S.A.
M. Mark Colavita, Christian Lindensmith, Chris Shelton, Mitchell Troy, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. 01109, U.S.A.
Peter M. Thompson, Systems Technology, Inc., Hawthorne, CA. 90250, U.S.A.

Abstract

The primary (M1), secondary (M2) and tertiary (M3) mirrors of the Thirty Meter Telescope (TMT), taken together, have over 10,000 degrees of freedom. The vast majority of these are associated with the 492 individual primary mirror segments. The individual segments are converted into the equivalent of a monolithic thirty meter primary mirror via the Alignment and Phasing System (APS) and the Primary Mirror Control System (M1CS).

In this paper we first provide an introduction to the TMT. We then describe the overall optical alignment and control strategy for the TMT and follow up with additional descriptions of the M1CS and the APS. We conclude with a short description of the TMT error budget process and provide an example of error allocation and predicted performance for wind induced segment jitter.

INTRODUCTION

The Thirty Meter Telescope (TMT) is a collaborative project between the California Institute of Technology, the University of California, the Association of Canadian Universities for Research in Astronomy, the National Astronomical Observatory of Japan, the Department of Science and Technology of India, and the National Astronomical Observatory of China.

The TMT design is a f/15, wide-field, altitude over azimuth, Ritchey-Chretien telescope with a 30 m primary mirror composed of 492 hexagonal segments. The telescope when pointing at zenith is ~ 51 m high and weighs approximately 1800 metric tons.

The tertiary mirror is articulated and in combination with the large Nasmyth platforms enables the mounting of eight or more different AO/instrument combinations. The telescope will support observations from 0.31 to 28 μm. The TMT will include integrated advanced adaptive optics capabilities, including a laser guide star system, supporting diffraction-limited observations at wavelengths beyond 1 μm over most of the sky. The wide, 20 arc-minute diameter, field-of-view will enable the use of wide-field, multi-object spectrographs. The "early light instruments" that are delivered as part of the construction effort include IRIS (Infrared Imaging Spectrometer), WFOS (Wide Field Optcal Spectrometer) and IRMS (InfraRed Multi-slit Spectrometer).

TMT will be sited at Mauna Kea, Hawaii. Construction of the telescope is scheduled to begin in 2014 with first light with all 492 segments in 2021.

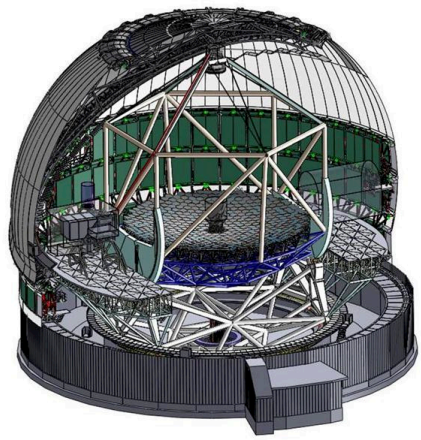


Figure 1: The Thirty Meter Telescope.

CONTROL SYSTEM OVERVIEW

The TMT image quality control architecture can be decomposed into two major systems; Active and Adaptive Optics. Active Optics (aO) is accomplished by the three mirror telescope and is responsible for the image quality of the optical beam delivered to the seeing limited science instruments or the Adaptive Optics system. The Adaptive Optics (AO) system is responsible for delivering diffraction limited image quality to the infrared instruments by attenuating the blurring effects of the atmosphere, reducing image jitter induced by the telescope drives and wind shake, and reducing residual image quality errors in the beam delivered by the aO system. The remainder of this paper is focused on the aO system.

The aO system maintains TMT image quality with a total of 11,815 degrees of freedom distributed across four principle local control loops; the Mount Control System [1], the M1CS, the M2 Control System (M2CS), and the M3 Control System (M3CS). Each of the principle control loops, with the exception of the M3CS, takes advantage of real time corrections based on on-sky measurements provided by an outer control loop. The on-sky corrections can come from one of three sources; the APS,

an Acquisition, Guider and Wave-Front Sensor (AGWFS) system located in each of the seeing limited instruments [2], or offloads from the AO system. The APS is used only when aligning the telescope optics and calibrating the sensors associated with the principle local control loops whereas the other two sources of correction are used during science observations.

An accounting of the degrees of freedom along with a listing of the key characteristics of each principle local control loop is tabulated in Table 1. Table 1 also includes a description of the relationship between each of the principle local control systems and the APS and AGWFS outer control loops used during alignment and science observations respectively.

Table 1: Characteristics of M1, M2, M3, and Outer Control Loops

Principle Local Control Loops							Alignment and Calibration Loop		Operational Outer Loop	
Name		Degrees of Freedom	Actuators	Sensors	Update Rate (Hz)	Loop BW (Hz)	Sensor	Refresh Rate	Sensor	Update Rate (Hz)
Mount	Azimuth & Elevation	2	Direct Drive	Tape encoders	≥ 40	~ 1	APS camera	Monthly	AGWFS (Guider)	1
M1	Global Tip, Tilt, Piston	3	Segment actuators	Actuator sensors	≥ 10	~ 1	Surveying/FEM	> 1 year	No outer control loop	
	Segment Tip, Tilt, Piston	1476	Segment actuators	Edge sensors	≥ 10	~ 1	APS	2 to 4 weeks	AGWFS (WFS)	0.003
	Warping Harness	10,332	Warping harness	Strain gauges	Set & Forget	na	APS	2 to 4 weeks	No outer control loop	
M2	De-center	2	Hexapod	Local encoders	≥ 10	< 1	APS	2 to 4 weeks	AGWFS (WFS)	0.003
	Tip/Tilt	2	Hexapod	Local encoder	≥ 10	< 1	Surveying	> 1 year	No outer control loop	
	Piston	1	Hexapod	Local encoder	≥ 10	< 1	APS	2 to 4 weeks	AGWFS (WFS)	0.003
M3	Tilt	1	DC drive	Local encoder	≥ 10	< 1	APS (Pupil Tracker)	> 1 year	No outer control loop	
	Rotation	1	DC drive	Local encoder	≥ 10	< 1	APS (Pupil Tracker)	> 1year	No outer control loop	

For simplicity we can describe the TMT aO image quality control strategy using five objectives. Each objective is accomplished using one or more of the four principle local control loops defined previously in combination with the APS or the AGWFS. The five objectives are M1 global shape, M1 Segment Shape, Alignment, Acquisition and Pointing, and Guiding. In practice these objectives are coupled but for the purposes of this paper we assume they are independent. Although Acquisition and Pointing is not, strictly speaking associated with image quality, it is included here for completeness. Each objective is briefly described below.

M1 Global Shape

The APS is used on-sky to determine the 2772 M1 edge sensor readings that result in the formation of the equivalent of a 30 m monolithic mirror. The APS achieves this by accurately measuring, and then positioning each of the 492 segments in piston, tip, and tilt. Once the APS determines that the global shape of the M1 is correct the corresponding edge sensor readings are recorded for later use by the M1CS.

In operation the M1CS will use the edge sensor readings recorded by the APS to maintain the overall shape of the M1 despite structural deformations caused by temperature and gravity, and disturbances from wind and vibrations. In addition the M1CS will receive low temporal and spatial frequency real time corrections from the AGWFS. These corrections reduce residual errors in the desired edge sensor readings as well as attenuate temporal and thermal drifts in the M1, M2, and M3 shapes.

M1 Segment Shape

The individual segments will have shape errors associated with polishing, coating stresses, and the support system. In addition, there will be segment shape errors associated with segment positions in the telescope deviating from their ideal positions.

The shape of each segment can be adjusted via a 21-actuator warping harnesses that is built into the support assembly for each segment. The APS measures and partially corrects, via the M1CS, the shape of each of the 492 segments using the 21 warping harness actuators. These corrections are typically made after a segment exchange and held constant until the next exchange. Ten segments are exchanged every two weeks for re-coating.

Alignment

The optical axis of the telescope is defined by the global position of the M1. The global position (piston, tip, tilt) of the segmented M1 is allowed to systematically follow the deformations of the M1 support structure due to gravity as the telescope rotates about the elevation axis in a manner that minimizes the maximum stroke of the 1472 segment actuators. To achieve this result the desired global position of the M1 as a function of elevation angle is determined via analysis of the telescope structure Finite

Element Model (FEM). Real time measurement of the global M1 position in operation is achieved by determining the best fit plane through M1 as determined by the actuator sensors

In operation the M2 is constantly re-positioned in translation to maintain alignment with the M1 and in piston to keep the optical system in focus. The proper position of the M2, as a function of telescope elevation angle, is determined on-sky by the APS. In operation the M2CS will use the translations recorded by the APS to maintain proper alignment with the M1. In addition the M2CS will receive real time corrections from the AGWFS's on-sky measurements. These corrections reduce residual errors in the M2 position determined by the APS and correct thermal and temporal drifts in the M1 and M2 positions and M1-M2 de-space.

Acquisition and Pointing

Acquisition and Pointing is the process of positioning the telescope, without on-sky feedback, so that the desired object is at the correct location on the science detector. As described above, the M1 defines the optical axis and wanders systematically with elevation angle. In addition the M3 needs to be properly positioned, as a function of telescope elevation angle, to direct the optical beam to the correct instrument

Meeting the required one arc-second RMS pointing error requirement over the full sky requires an accurate model of the non-ideal telescope structure behaviour. On a monthly basis pointing models are built using data collected with the APS acquisition camera. The pointing model is used to correct systematic structural deformations, axis misalignments, mount encoding errors, M1 motion relative to the azimuth and elevation axes, and systematic errors in the positioning of M3.

Guiding

After acquiring an object the telescope mount and the M3 must move in a coordinated fashion to remove the effects of earth's rotation and maintain the object in the correct location on the science detector. This is accomplished using the pointing models described above complemented with real time optical feedback via the AGWFS. The AGWFS is mounted in the field of view of the instrument and measures the position of a bright object relative to the nearby science object. The corrections from the AGWFS are used to maintain the science image at the correct position by correcting residual low temporal frequency pointing model errors and thermal and temporal drifts. The corrections are applied to the Mount Control System.

The M3 does not receive real-time optical feedback. Since there is degeneracy between the articulated M3 drive and the mount elevation drive, image position errors are corrected on the mount while the M3 is driven to maintain pupil alignement. This requires that the M3 rotation and tilt axes to be well calibrated.

M1CS

The M1CS is responsible for maintaining the overall shape of the segmented M1 mirror over all conditions. Properly supported, the mirror segments can be treated as rigid bodies; hence, their positions can be described by six parameters. The three in-plane motions are controlled passively via the Segment Support Assembly [3]. The three out-of-plane motions (piston, tip, tilt) are actively controlled by the M1CS via three actuators per segment and two sensors per inter-segment edge. In total the M1CS contains 1476 actuators and 2772 sensors. The M1CS also provides local control of the warping harnesses used to adjust segment shape.

The M1CS can be considered a stabilization system that works to maintain the shape of M1 based on previously determined set-points. The set-points vary as a function of zenith angle and temperature.

The starting point for the design of the TMT M1CS is the M1 control system used on the successful 10 m Keck telescopes [4]. The TMT actuators and sensors are technically challenging as a result of their aggressive cost target, and performance and reliability requirements.

The two sensors located on each inter-segment edge measure a linear combination of height difference and the dihedral angle between adjacent segments as well as providing a separate measurement of the gap between adjacent segments. The difference between the sensor measurements and the desired sensor readings determined by APS are processed by the control algorithm at a 20Hz rate to determine the actuator commands necessary to maintain the correct shape of the primary mirror. The TMT and Keck edge sensors are capacitive but the TMT design diverges significantly from the Keck design in detail. The TMT design is simpler, cost effective, and eases the effort associated with segment exchanges [5]. These benefits come with a price though; the TMT sensor is more sensitive to in-plane motions of the segments and hence requires a innovative calibration scheme.

The actuator design is based on a parallel combination of a 10 Hz control loop bandwidth closed on a voice coil using an accurate commercially available optical sensor for fine positioning and an offload loop that works to minimize the power dissipation in the voice coil. The offload loop provides the quasi-static force required to support the weight of the mirror at all zenith angles with near zero power dissipation. The voice coil control laws create equivalent stiffness and damping properties that achieve high positioning accuracy even in the presence of external disturbances such as wind [6].

In addition to the technology challenges above the TMT control design is significantly more challenging than what was required at Keck. TMT requires a 1 Hz global bandwidth for wind rejection compared to the 0.1Hz bandwidth at Keck. The required 1 Hz bandwidth coupled with the additional compliance resulting from TMT's much larger structure results in a significant Control Structure Interaction challenge which is dealt with by clever control design and adding damping to the actuators. In addition, uncertainty in the control matrix

requires significant care in determining the appropriate margins on the control design [7, 8].

ALIGNMENT AND PHASING

The APS uses a Shack-Hartmann wave-front sensor to measure and correct the shapes of the individual segments and the overall image quality of the telescope and a phasing camera to phase the individual segments of the M1. APS achieves the required image quality by using on-sky measurements and adjusting segment piston, tip, and tilt, segment surface figure via warping harness adjustments, and the rigid body degrees of freedom of M2 and M3. APS will be used after segment exchanges.

The design of the APS is based on the Phasing Camera System (PCS) used at the Keck telescopes. The alignment process and algorithms that were developed for Keck are directly applicable to TMT [9].

There are some critical differences and challenges when scaling up PCS to the APS. In both the PCS and APS, segment piston, tip, and tilt corrections are accomplished in parallel. On the other hand measurement and correction of segment shapes is a serial process at Keck whereas it will be a parallel process at TMT. In addition, at Keck the adjustment of the warping harnesses is accomplished manually in the mirror cell the day after the PCS measurements were taken. At TMT the corrections are automated allowing for efficient iteration and convergence; and therefore will be performed in the early evening of the same day as the segment exchange.

PERFORMANCE, REQUIREMENTS AND PREDICTION

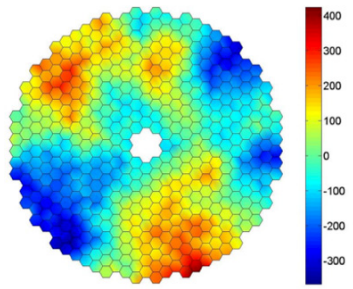
TMT is using a formalized system engineering approach to the development and flow-down of performance requirements. In addition TMT has developed and is utilizing a new metric for characterizing the seeing-limited performance of large telescopes, the normalized point source sensitivity (PSSN) [10]. In PSSN space unity means no performance degradation relative to a perfect telescope operating under identical atmospheric conditions. The overall PSSN requirement for TMT is 0.85 which is equivalent to saying that the telescope performance cannot degrade relative to the perfect telescope operating in a similar atmosphere by more than 15%.

A comprehensive description of TMT performance requirements and prediction is beyond the scope of this paper. Instead, we provide an example where we discuss the flow down of the top level requirement on segment dynamic displacement residuals to the allocation on wind driven segment motion and illustrate the corresponding predicted performance.

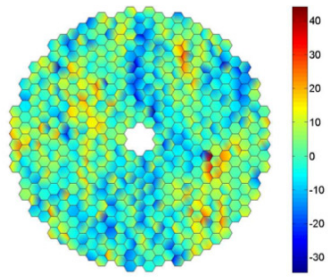
A tabulation of the errors sources and PSSN allocations is shown in Table 2. The allocation for wind driven segment motion is 0.994. Figure 2 illustrates the amount of segment jitter induced under mean wind conditions with the M1CS on and off. The results indicate that the performance meets the allocated PSSN error.

Table 2: PSSN Error Allocation for Segment Dynamic Displacement Residuals

Segment dynamic displacement residuals	**0.989**
Edge Sensor noise	0.996
Actuator noise	0.999
Segment wind buffeting	*0.994*
Segment micro-seismic vibration	0.999
Segment equipment vibration	0.999



M1CS Off, 127 nm RMS, PSSN = 0.956



M1CS On, 7.5 nm RMS, PSSN = 0.994

Figure 2: Comparison of wind driven segment motion with M1CS off and on.

REFERENCES

[1] Peter M. Thompson, "*Analysis of the TMT mount control system*", SPIE Vol. 7012, July 2008

[2] Carl Nissly, et al, "*Investigation of Thirty Meter Telescope Wavefront Maintenance Using Low Order Shack-Hartmann Wavefront Sensors to Correct for Thermally Induced Misalignments*", SPIE Vol. 7738, July 2010

[3] Eric C. Williams, et al, "*Advancement of the segment support system for the Thirty Meter Telescope primary mirror*", SPIE Vol. 7018, July 2008

[4] R.C. Jared, et al, "*The W. M. Keck Telescope segmented primary mirror active control system*", SPIE Vol. 1236, (1990)

[5] Chris Shelton, et al, "*Advances in edge sensors for the Thirty Meter Telescope primary mirro*r" Proceedings of SPIE Vol. 7012, July 2008

[6] Peter M. Thompson, et al, "*Servo design and analysis for the Thirty Meter Telescope primary mirror actuators*", SPIE Vol. 7733, August 2011

[7] Douglas G. MacMynowski, et al, "*Robustness of Thirty Meter Telescope primary mirror control*", SPIE Vol. 7733, August 2010

[8] Martin W. Regehr, et al "*Dynamic characterization of a prototype of the Thirty Meter Telescope primary segment assembly*", SPIE Vol. 7733, 5 August 2010

[9] Gary Chanan, et al, "*W.M. Keck Telescope phasing camera system*" SPIE Vol. 2198, June 1994

[10] Byoung-Joon Seo, et al, "*Analysis of normalized point source sensitivity as a performance metric for large telescopes*", Applied Optics, 48(31), 5997-6007 (2009)

A DIGITAL SYSTEM FOR LONGITUDINAL EMITTANCE BLOW-UP IN THE LHC

M. Jaussi*, M. E. Angoletta, P. Baudrenghien, A. Butterworth,
J. Sanchez-Quesada, E. Shaposhnikova, J. Tuckmantel,
CERN, Geneva, Switzerland

Abstract

In order to preserve beam stability with nominal bunch intensity in the LHC, longitudinal emittance blow-up is performed during the energy ramp by injecting phase noise in the main accelerating cavities. The noise spectrum spans a small frequency band around the synchrotron frequency. It is generated continuously in software and streamed digitally into the Digital Signal Processor (DSP) of the Beam Control system where it is added to the pick-up signal of the beam phase loop, resulting in a phase modulation of the accelerating RF. In order to achieve reproducible results, a feedback system, using as input the measured bunch lengths averaged over each ring, controls the strength of the excitation, allowing the operator to simply set a target bunch length. The spectrum of the noise is adjusted to excite the core of the bunch only, extending to the desired bunch length. As it must follow the evolution of the synchrotron frequency through the ramp, it is automatically calculated by the LHC settings management software from the momentum ramp and RF voltage. The system is routinely used in LHC operation since June 2010. We present here the details of the implementation in software, FPGA firmware and DSP code, as well as some results with beam.

INTRODUCTION

Effective control of the bunch length in the LHC is essential to avoid loss of Landau damping leading to longitudinal instability during the energy ramp [1]. A longitudinal emittance blow-up system has been developed, inspired by the system that has been used in the SPS for several years [2]. The beam is excited with RF phase noise acting on the fundamental RF system (400.8 MHz). The frequency spectrum of the noise is tailored to the synchrotron frequency spread of the particles in the bunch so as to selectively excite a chosen portion of the particles centered around the core of the bunch [3]. The frequency of a single-particle synchrotron oscillation depends on the peak amplitude of its trajectory ϕ_{pk}

$$\Omega_s(\phi_{pk}) \approx \Omega_{s0}\left[1-\left(\frac{\phi_{pk}}{4}\right)^2\right] \qquad (1)$$

with Ω_{s0} the synchrotron frequency of the zero-amplitude oscillation. For $\phi_{pk} = \frac{\pi}{2}$ the synchrotron frequency is $\sim \frac{6}{7}\Omega_{s0}$. A rectangular frequency spectrum extending from $\frac{6}{7}\Omega_{s0}$ to Ω_{s0} will excite particles in a 1.25 ns window of the 400 MHz bucket. We extend the spectrum to $1.1\Omega_{s0}$ to guarantee that we do not miss the core. The noise is injected into the RF via the beam phase loop, whose main purpose is to keep the accelerating RF in phase with the beam [4]. The noise signal is added digitally into the phase loop error signal, "shaking" the phase of the accelerating cavity voltage with respect to the beam to produce the desired excitation.

* michael.jaussi@cern.ch

REQUIREMENTS

The system needs to generate band-limited noise that must follow the synchrotron frequency and with an amplitude that eventually achieves the desired bunch length. It must be able to run for a long period of time, since the ramping process in LHC lasts for 11 minutes (it was 40 minutes in 2010). The system must allow great flexibility in the noise generated as it was not well known at the commissioning of the blow-up process what would be the most effective settings for the beam, and how to achieve the targeted bunch length.

IMPLEMENTATION

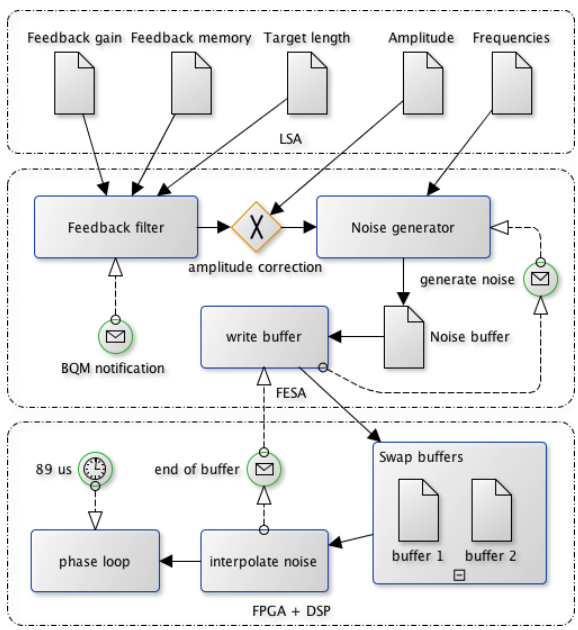


Figure 1: Block diagram showing the 3 layers.

The system is subdivided into 3 layers (see Fig. 1). From the bottom up, we have the hardware layer, implemented

in a VME board, containing a Field Programmable Gate Array (FPGA) and an Analog Devices TigerSHARC DSP, responsible for the digital processing of the beam control loops. In the CPU of the VME crate we have the device driver and the front-end software, written in C++ under CERN's Front-End Software Architecture (FESA) [5] framework. Finally we have the control system application software based on LSA [6], and written in Java, which handles settings management for the whole accelerator.

FPGA AND DSP

The FPGA and DSP are in the phase loop module of the LHC beam control. The FPGA implements a double-buffering scheme to allow continuous operation for the DSP. It provides 2 buffers of 4096 float values which are alternately filled via the VME bus by the front-end software. The FPGA presents the data one point at a time to the DSP which injects it into the phase error correction. Once a buffer is empty, the FPGA swaps the buffers and sends an interrupt to the CPU asking for the empty buffer to be filled.

The start of the noise generation is triggered from the software by writing a bit in the appropriate register. The processing rate of the phase loop is once per machine turn, so the DSP has to calculate a new value about once every 89 μs. With a synchrotron frequency varying between 60 Hz and 28 Hz during the acceleration ramp, the spectrum of the excitation data does never extend beyond 70 Hz. The DSP code specifies the rate at which the buffer should be read and intermediate samples are interpolated to achieve the 89 μs period. In the buffer, the noise samples are at the 2 kHz rate (one value every 500 μs.) so with a buffer size of 4096, we need to fill a new buffer every 2 seconds. The FPGA has a few registers for buffer handling. This allows us to set the interpolation rate, specify the buffer size (up to 4096), enable the interrupt and flag a buffer as ready. A flag is also raised when a buffer is empty and a new buffer is required.

FESA CLASS

To control the FPGA and DSP, the FESA class accepts settings from the control system and then generates the noise. The settings are :

- 2 functions for programming the upper and lower frequency during the ramp.
- 1 function for programming the maximum amplitude allowed.
- 2 functions programming for the feedback system.
- 1 scalar or function for programming the target bunch length.
- A way to start the system, either manually or through a trigger.

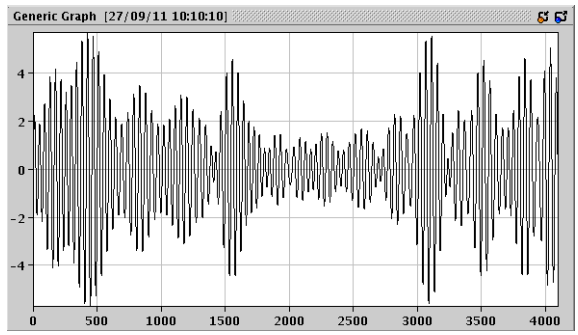


Figure 2: Noise buffer calculated by the FESA class.

As the noise generated cannot fit entirely in memory, it is generated in partial records (example Fig. 2), and we keep track of the position in time within the functions by counting the number of noise samples generated. This gives us a time precision equal to 500 μs with the current settings, which is more than good enough as the frequencies variy slowly (see Fig. 4). In order to react quickly to the interrupt sent by the FPGA when a buffer is empty, one buffer of noise is always generated in advance. When the interrupt is sent, it triggers a FESA real-time task which writes the noise data into the buffer and then acknowledges it. This task then fires a user-event which triggers another real-time task with a lower priority in charge of generating the next noise buffer. If any of the settings was updated, either by the feedback or by human intervention (such as changing a function), then the new values are taken into account at this point of time. If for some reason, the noise cannot be generated before the next interrupt, an alarm is raised and the system set the noise output to zero.

Noise Generation Algorithm

To allow the generation of precisely band-limited noise, a dedicated noise generation algorithm was developed [7]. It uses sine and cosine generators which are initialized to a random phase, scaled with a random amplitude and that generate noise in a normalized frequency range, with the generators frequencies equally spaced between 0 and 1. The noise is then translated numerically to the correct frequency range, defined by two parameters, named f_{up} and f_{low} in [7], which reflect the upper and lower bound of the range. To allow different spectral shapes, there is a weighting of all generators amplitudes, allowing for the generation of any spectrum, with a resolution depending on the number of generators. Another parameter specifies the RMS amplitude of the noise. To modulate the amplitude, each value is simply multiplied by the desired scaling.

Feedback

The feedback system is divided into two blocks. The first one is the measurement and the second one is the correction of the excitation by continuously adjusting the amplitude of the phase noise. It must be noted that, because of the next

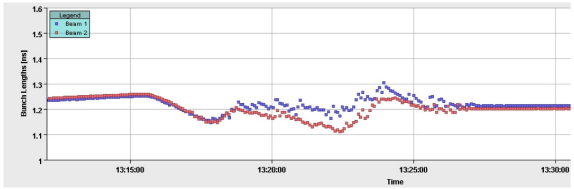


Figure 3: Four-sigma bunch length evolution during a ramp with the blow-up system set for a 1.2 ns target. Start of ramp at ∼13:16, end at ∼13:27

buffer already generated in memory, the change induced by the feedback is only taken into account for the next buffer, which might be used only after 2 to 4 seconds. This could be reduced by using smaller buffer size, but then a new noise buffer must be provided more often, inducing a bit more overhead on the CPU as more interrupts need to be handled.

Bunch length measurement: In order to see the result of the noise injected, we need to measure the length of each bunch (or at least have a mean length over all bunches). To do so, we use an existing diagnostic system called the Beam Quality Measurement (BQM) [8], which uses high-speed digitizers to acquire the signal from a wideband longitudinal beam pickup over a complete turn of the machine and then calculates from the signal a length for each bunch (see Fig. 3). The calculation extracts the Full Width at Half Maximum (FWHM) for each bunch, and estimates the 4σ equivalent length assuming a Gaussian profile. As this system is also implemented using a FESA class, it allows other systems to subscribe to the measurement, thus allowing our system to be notified each time a new measurement is available. The measurement rate of this system is about 5 seconds at the time of writing, which is rather slow, and is in fact one of the major limitations on performance of the blow-up. A upgrade is underway to improve the data rate of this system.

Filter: The feedback system is a simple low pass filter, with two variable parameters, defined as functions of time to allow fine tuning of the system. One parameter is the gain, which is increased during the ramp to compensate for the reduced sensitivity to RF noise with energy. The second one is the time constant of the filter, intended to smooth the bunch length measurements.

CONTROL SYSTEM

The control system provides a way to store, generate and send the settings to the FESA class. Most of the settings are defined as functions which vary along the ramp. Some settings are declared as being dependent on other settings. This is the case for the frequencies as they are calculated as a band around the synchrotron frequency (Fig. 4). This frequency is in turn dependent on the momentum and the RF voltage. The target bunch length (Fig. 5) is typically set

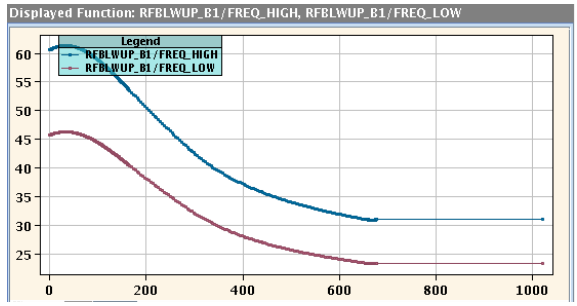


Figure 4: Upper and lower frequency bound functions in LSA. Horizontal axis: time in ramp (sec), vertical axis: frequency (Hz).

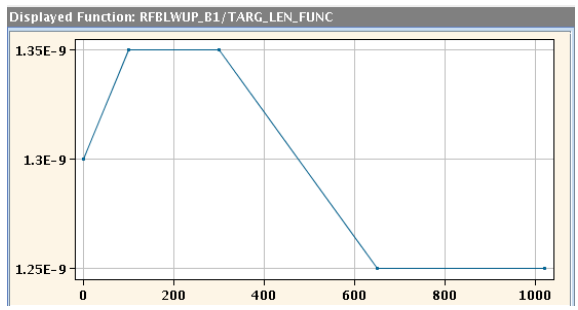


Figure 5: Target bunch length function in LSA.

to a higher value at the beginning of the ramp to mitigate the shrinking which occurs due to the increase of the RF voltage. The feedback gain (Fig. 6) is increased during the ramp as the beam becomes less responsive to the excitation as the energy increases. The control system is in charge of updating the frequency values if any of the dependent parameters changes. It is also responsible for arming the system before the ramp, allowing it to start on reception of an event from the accelerator timing system. The arming process is needed as the event used as a trigger is far from unique and could happen also when the machine stays at injection energy. Instead of having a dedicated timing, the sequencer software that prepares the entire ramp process takes care of arming the trigger. The control system also monitors any alarm that could be raised by the FESA class.

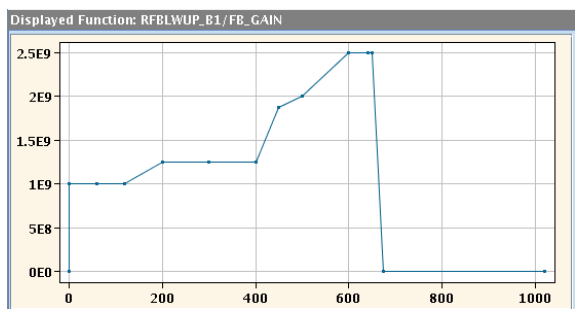


Figure 6: Gain function in LS.

RESULTS AND CONCLUSIONS

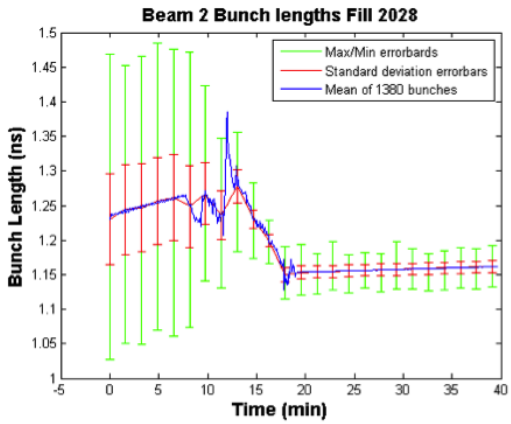


Figure 7: Statistics on bunch length (mean, min, max and standard deviation error bars) measured by the BQM during the ramp.

The first results without feedback, for a few bunches, were encouraging. It demonstrated that the system was working, but also that a feedback system was required to ensure reproducibility of the results. Once the feedback system was in place, the results were seen to be highly reproducible, and the system has been in use operationally since June 2010. A good feature of the blow-up is that it reduces the spread of bunch lengths. As can be seen in Fig. 7, the spread of bunch length at injection is large, with a standard deviation of about 60 ps, which after the ramp is reduced to around 15 ps. We often observe periods of very fast change in the measured bunch length, and this limits the final accuracy of bunch length control. We believe that these transients come from a change in bunch profile that has a significant impact on the FWHM measurement without much increase in longitudinal emittance. This has an impact on the stability of the feedback and sometimes results in missing the target bunch length at the end of the ramp. To mitigate this type of problem, the BQM system is being updated to provide measurements at a higher rate, which will allow higher feedback gain settings without becoming unstable.

REFERENCES

[1] E. Shaposhnikova *et al.*, "Loss of Landau damping in the LHC", IPAC'11, San Sebastian, Spain, 4–9 September 2011,

[2] J. Tuckmantel *et al.*, "Study of Controlled Longitudinal Emittance Blow-up for High Intensity LHC beams in the CERN SPS", EPAC08, Genoa, Italy, June 2008,

[3] P. Baudrenghien *et al.*, "Longitudinal emittance blow-up in the LHC", IPAC'11, San Sebastian, Spain, 4–9 September 2011,

[4] P. Baudrenghien *et al.*, "The LHC Low Level RF", EPAC06, Edinburgh, UK, June 2006,

[5] M. Arruat *et al.*, "Front-End Software Architecture", ICALEPCS07, Knoxville, USA, October 2007,

[6] G. Kruk *et al.*, "LHC Software Architecture [LSA]: Evolution toward LHC Beam Commissioning", ICALEPCS07, Knoxville, USA, October 2007,

[7] J. Tuckmantel, "Digital Generation of Noise-Signals with Arbitrary Constant or Time-Varying Spectra", LHC Project Report 1055, February 2008,

[8] G. Papotti *et al.*, "Longitudinal Beam Measurements at the LHC: the LHC Beam Quality Monitor", IPAC'11, San Sebastian, Spain, 4–9 September 2011,

OPERATIONAL STATUS OF THE TRANSVERSE MULTIBUNCH FEEDBACK SYSTEM AT DIAMOND

I. Uzun, M. Abbott, M.T. Heron, A.F.D. Morgan, G. Rehm
Diamond Light Source, Oxfordshire, UK

Abstract

A transverse multibunch feedback (TMBF) system is in operation at Diamond Light Source to damp coupled-bunch instabilities up to 250 MHz in both the vertical and horizontal planes. It comprises an in-house designed and built analogue front end combined with a Libera Bunch-by-Bunch feedback processor and output stripline kickers. FPGA-based feedback electronics is used to implement several diagnostic features in addition to the basic feedback functionality. This paper reports on the current operational status of the TMBF system along with its characteristics. Also discussed are operational diagnostic functionalities including continuous measurement of the betatron tune and chromaticity.

INTRODUCTION

The 3 GeV Diamond storage ring is currently serving photon beam for users with a current of 250 mA in a multibunch fill of 900 bunches. In addition, a hybrid mode where 3/4 of the ring filled with a multibunch and a high intensity single bunch sitting on the opposite side is run for certain experiments. Multibunch instabilities have been clearly observed and studied at Diamond from the early days of commissioning [1]. Indeed the operation with the nominal zero chromaticity lattice is prevented at multibunch currents as low as 10 mA in a uniform fill with 936 bunches. Therefore, it was necessary to create a transverse multibunch feedback system to combat the instabilities towards achieving our final target current of 300mA in multibunch mode.

OVERVIEW OF THE TMBF DESIGN

The Diamond TMBF System is a wideband feedback correcting the positions of individual bunches, spaced 2ns apart. Figure 1 shows the system level structure of the TMBF system. It is composed of a detector in terms of pickup buttons, hybrids (Figure 2) and an in-house developed RF front end (Figure 3) inspired by an existing ESRF design, a commercial digital FPGA based feedback electronics called Libera Bunch-By-Bunch from Instrumentation Technologies [2], stripline kickers and 100W power amplifiers.

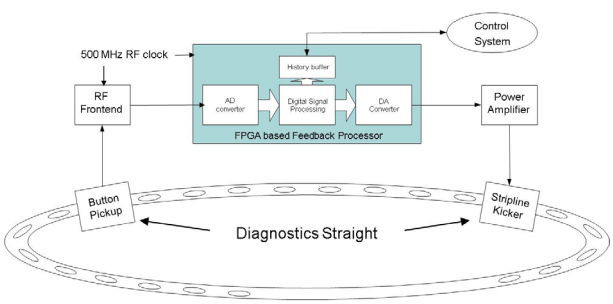


Figure 1: Block diagram of the Diamond TMBF system.

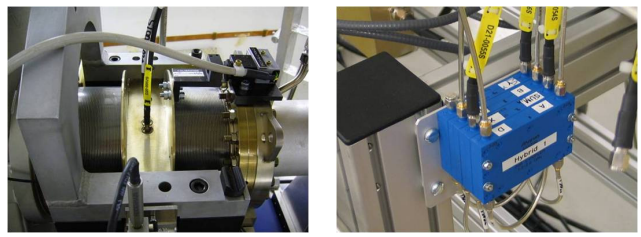


Figure 2: 4-button pickups and hybrids installations in the ring.

Wideband position signals are picked up by a set of four button pickups, and then passed through a set of hybrid combiners to generate the X and Y position signals. These wideband signals are passed to an RF front end, where the signal is mixed and filtered to 0-250 MHz baseband signal and then amplified to the correct level for the electronics.

The position signal is directly sampled at the RF frequency with 4 ADCs (f_{rf} = 500MHz and $f_{adc} = f_{rf}/4$). On the basis of the betatron motion measured for each bunch, FPGA-based electronics performs FIR (Finite Impulse Response) filtering to feedback the signal

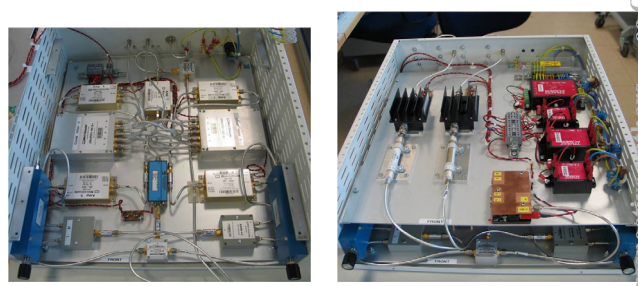


Figure 3: In-house developed RF front end occupies a 3U 19" rack and built over two layers. The bottom layer (left) houses the input comb filters and the mixer, while the the timing module, output filters, splitters and power supplies are located on the upper layer (right).

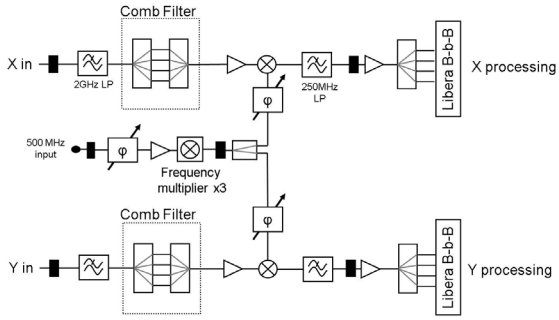


Figure 4: RF front end schematics. A phase shifter was added on the 500MHz RF clock input path to compensate electron beam phase shifts relative to the RF clock.

necessary for oscillation damping of a given bunch turn by turn. This information is passed to a power amplifier which, through some low pass filters, drives two sets of kicker striplines differentially (one in X and one in Y) to apply the corrections to the beam.

The details of the TMBF system and its FPGA-based digital feedback processing capability have previously been discussed [3, 4]. Since our last status report [5], the main modification on the system was the addition of a phase shifter operating at 500MHz to the local oscillator chain in the front end electronics as shown in Figure 4. This was required as the RF phase of the synchrotron drifts slowly relative to the 500MHz timing signal. This has the effect of moving the sampling point on the bunch. If the phase shift is too large, the sample point moves off the bunch completely and the tune signal disappears. As the drift is very slow (on the order of days or weeks), this new phase shifter is not in a control loop; rather the operators can adjust the bunch intensity as necessary.

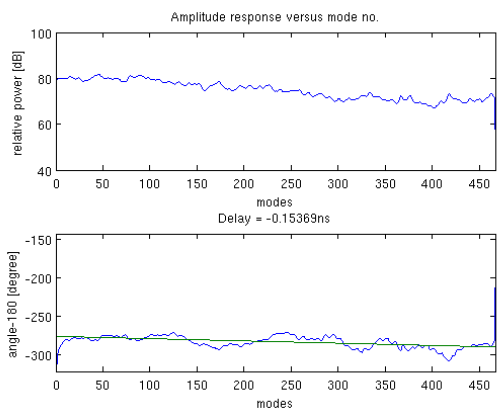


Figure 5: Variation of loop gain (top) and open loop phase response (bottom) as a function of mode number for the horizontal plane. The straight-line fit on the phase response reports 153ps of loop delay.

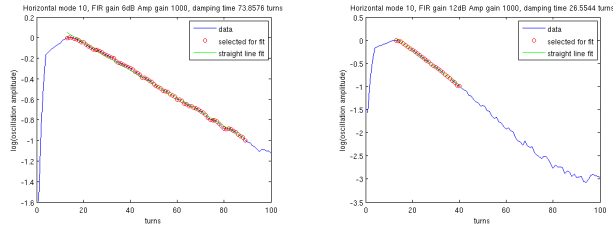


Figure 6: Damping time measurement for mode 10 in the horizontal plane. The fitting reports 73 and 25 turns damping time at FIR gains 0dB and 12dB, respectively.

PERFORMANCE

The stability of the feedback loop for all 936 modes is studied by exciting the beam at each mode frequency in turn using the internal Numerically Controlled Oscillator (NCO) and measuring the amplitude and phase response of the open loop at the output of the feedback FIR filter as shown in Figure 5. A straight-line fit on the phase response gives the overall loop delay, which guarantees a successful loop closure in terms of feeding back on to the right bunch in the train. The loop delay is tuned in 2ns steps in the FPGA, whilstt further tuning within the 2ns is achieved by modifying cable lengths in the loop.

To characterise the TMBF system, the feedback loop can be opened and closed momentarily in order to measure the growth and damping rate of individual modes. This measurement is carried out under the control of FPGA frequency sweep logic and internal timers. The routine to measure the damping time has been further automated to include post processing, including a straight-line fit on the decay. This produces a single number (the damping time) as an output. Figure 6 shows a typical damping time measurement.

Figures 7 shows a damping time scan for all modes in the horizontal plane at 250mA beam current with 2/3 fill and chromaticity set to 2. The damping time increases with rising mode number, as the gain of the feedback loop decreases with rising frequency. This is the conglomerate effect of the RF front end, the ADC, the drive amplifiers and the striplines. However, at low modes, where resistive wall effects drive instabilities more strongly, we achieve damping times of 73 in Horizontal and 82 turns in Vertical plane with FIR gain set to 0dB.

The feedback loop has been successfully closed to stabilise the beam up to 300mA with full fill, which is more challenging than the usual 2/3 fill. Figure 8 shows the transverse beam sizes as captured on a pinhole camera when the beam is stabilised by the feedback and when the beam is unstable without being lost.

In order to facilitate the measurement of the system characteristics and performance, a set of high-level graphical user interfaces has been developed using Matlab (Figure 9). It is our aim is to carry out these measurements routinely in order to understand the machine behaviour.

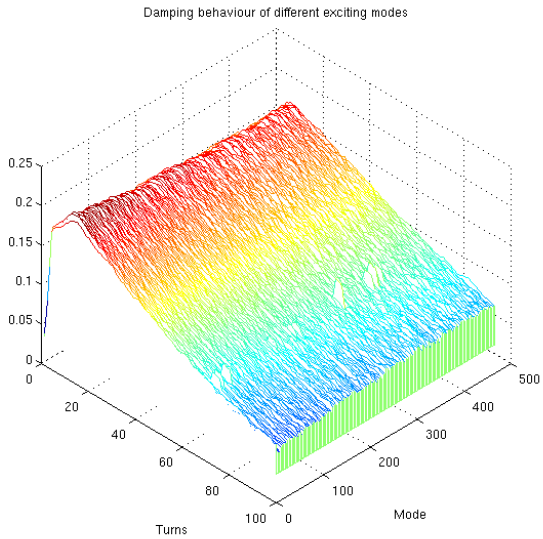


Figure 7: Damping time measurement for all instability modes in the horizontal plane.

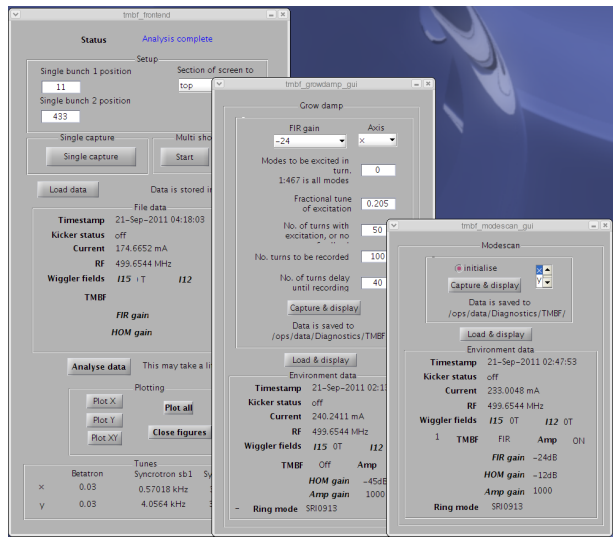


Figure 9: Matlab graphical user interfaces to launch diagnostic utilities. The main front end GUI for capturing and analysing raw data (left), grow/damp utility supporting measurement on multiple modes (middle), open-loop response measurement tool (right).

FPGA DIAGNOSTICS APPLICATIONS

The ability of the FPGA design to act on a selected single bunch in terms of applying feedback or excitation, or simply doing nothing, has enabled us to develop techniques for online tune and chromaticity measurements with no visible disturbance to user beam [6].

Measurement of continuous betatron tune with no visible disturbance to the user beam is achieved by exciting only a single bunch at a frequency near the nominal tune. The bunch motion is then detected by mixing the measured bunch position with a sine and cosine output of the excitation frequency and accumulating the result for a certain period over 100 turns. The excitation sweep is repeated for 4096 different frequencies with small increments and completed in less than a second. The result is delivered by the FPGA as waveforms of I and Q value per frequency point through the control system interface as shown in Figure 10.

The frequency-sweeping-based excitation for the tune measurement also produces information about the synchrotron oscillation sidebands to the tune. From the relative magnitudes of the tune sidebands, the chromaticity (tune shift with energy shift) can be derived. This application is currently in the commissioning phase, and is used at machine start-up when the chromaticity is assumed to be constant. It will subsequently be employed during user operation and allow us to replace an invasive measurement with a passive one giving a measurement of the chromaticity from the tune spectrum available every second. In doing so it will both reduce the impact on machine operations and improve our understanding of the behaviour during run time. Work-in-progress test results recorded on our stripline tool are shown in Figure 11.

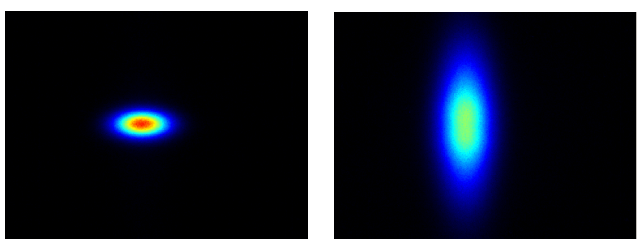


Figure 8: Pinhole screen captures of the electron beam with TMBF on (left) and TMBF off (right) at 250mA, 2/3 fill and chromaticity set to 0.

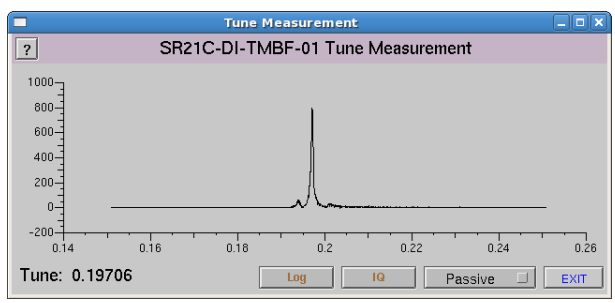


Figure 10: Online tune measurement (horizontal plane). The raw IQ data from FPGA can also available as Epics waveforms.

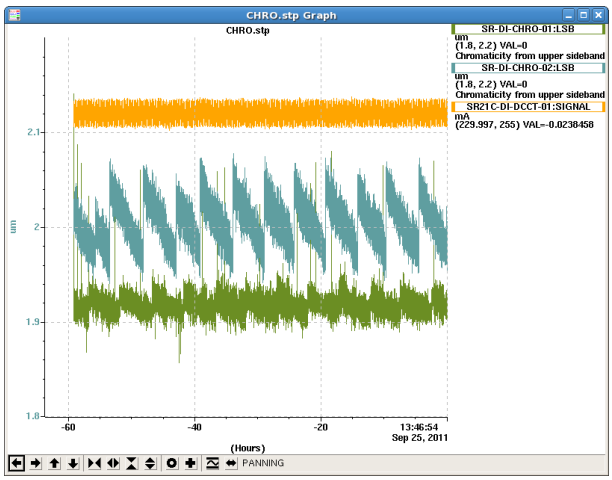


Figure 11: 60hrs of online acquired chromaticity data from the tune sidebands of a single bunch during top-up operation. Presently there is no bunch charge adjustment which explains the zig zag pattern.

CONTROL SYSTEM INTERFACE

The operation of the TMBF system is fully integrated into the Diamond EPICS control system. The user interface is divided into two modes of operation. The "Simple Mode" provides the higher level control interface in order to operate the TMBF system and reports basic status information such as feedback status, bunch amplitudes and the tune value (Figure 12 left). The "Expert Mode" provides a comprehensive control interface enabling access to all lower level control parameters on the FPGA (Figure 12 right). The expert mode is used mainly for commissioning and system debugging purposes.

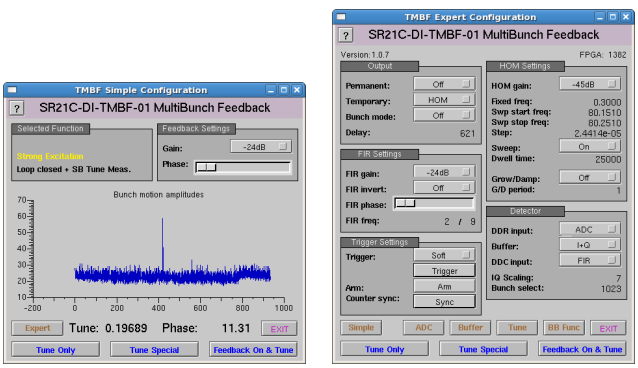


Figure 12: EPICS user interfaces. The selected function panel on the Simple Mode screen (left) shows that the feedback loop is closed and the single bunch tune measurement is activated as well. The spike on the bunch motion amplitude window confirms the excitation of a single bunch used for online tune measurement. The Expert Mode screen (right) provides access to all control registers.

CONCLUSIONS

The current status of the TMBF system at Diamond has been reported along with its performance. The feedback system has proven its ability to correct horizontal and vertical instabilities at 250mA with chromaticity set to 0. The feedback system is fully integrated into the EPICS control system for machine operation and supported by a useful set of Matlab applications for system analysis purposes.

REFERENCES

[1] R. Bartolini, et al. "Analysis of Collective Effects at the Diamond Storage Ring", EPAC'08, Genoa, June 2008.

[2] Instrumentation Technologies, "Libera Bunch-by-Bunch", http://www.i-tech.si/accelerators-instrumentation/libera-bunchbybunch/.

[3] A.F.D. Morgan, G. Rehm, I. Uzun, "First Tests of the Transverse Multibunch Feedback at Diamond", DIPAC'07, Venice, June 2007.

[4] I. Uzun, M. Heron, A. Morgan, G. Rehm, "An FPGA-Based Transverse Multibunch Feedback System For Diamond Light Source", FPL'10, Milano, September 2010.

[5] A.F.D. Morgan, G. Rehm, I. Uzun, "Performance and Features Of The Diamond TMBF System", EPAC'08, Genoa, Italy 2007.

[6] G. Rehm, et.all . "Measurement of Lattice Parameters Without Visible Disturbance to User Beam at Diamond Light Source", BIW'08, Santa Fe, New Mexica, May 2010.

USING TANGO FOR CONTROLLING A MICROFLUIDIC SYSTEM WITH AUTOMATIC IMAGE ANALYSIS AND DROPLET DETECTION

O. Taché, F. Malloggi
IRAMIS, LIONS, UMR SIS2M 3299 CEA-CNRS, CEA-Saclay, Gif-sur-Yvette Cedex, France

Abstract

The microfluidics is the science and technology of systems that process or manipulate small amounts of fluids, using channels with dimensions of tens to hundreds of micrometers. Since a decade microfluidics has become a powerful tool for fundamental and applied researches. Microfluidics influence subject areas from chemical synthesis and biological analysis to optics and information technology. At CEA / LIONS, we integrate microfluidics technology in several research projects. The present work deals with the development of tools for the detection and the analysis of complex calibrated micro-drops.

Although this technique uses small volume of chemicals, it requires the use of numbers of accurate electronic equipments such as motorized syringes, valve, pressure sensors and video cameras with fast frame rate coupled to microscopes.

We use a TANGO control system [1] for all these heterogeneous equipments in microfluidics experiments and video acquisition. We have developed a set of device servers which allow image acquisition and droplets detection (size, number, velocity) almost in real time.

Using TANGO, we are able to provide feedback to actuators, in order to adjust the size and the rate of the droplet formation.

MICROFLUIDIC PRINCIPLES

This technology is based on the manipulation of continuous liquid flow through microfabricated channels. In the lab we use rapid prototyping of polymer (polydimethylsiloxane-pdms) casting [2] and we designed original microfluidic chips for droplet formation.

The basic principle is the following. Two immiscible fluids (oil and water for instance) meet at a cross junction (see Figure 1a-b). Due to the viscous shear the liquid/liquid interface is deformed until a drop is created. In this particular case the local shear is well controlled leading to a very reproducible process [3]. The straightforward consequence is the formation of a monodisperse emulsion as it can be seen on the Figure 1c where a droplet crystal is shown.

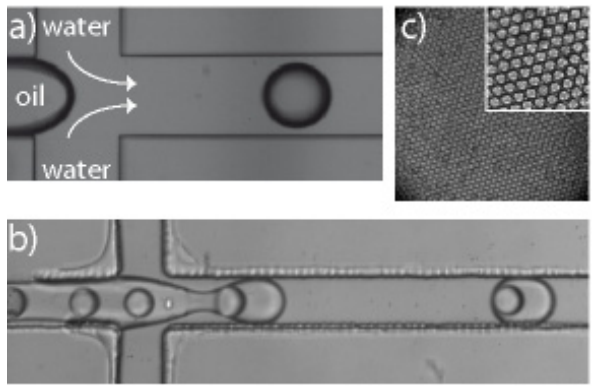


Figure 1: a- Droplet formation at a cross junction. b- Complex droplet formation at a cross junction (double emulsion). c- Crystal of droplets.

MICROFLUIDIC SYSTEM DESCRIPTION

The microfluidic system used here (see Figure 2) is composed by Nemesys [4] motorized syringes, an inverted microscope (Olympus IMT-2) with a x10 magnification objective and a fast video camera Phantom from Vision Research [5]. Syringes system and video camera are controlled by the computer. This combination gives a pixel size on images of 1.92µm.

For producing oil drops we use two syringes filled with water+surfactant and one filled with oil. Products are injected in a dedicated microfluidic chip designed in the laboratory. Flow rate can vary typically from 0.1µl/min to 50µl/min.

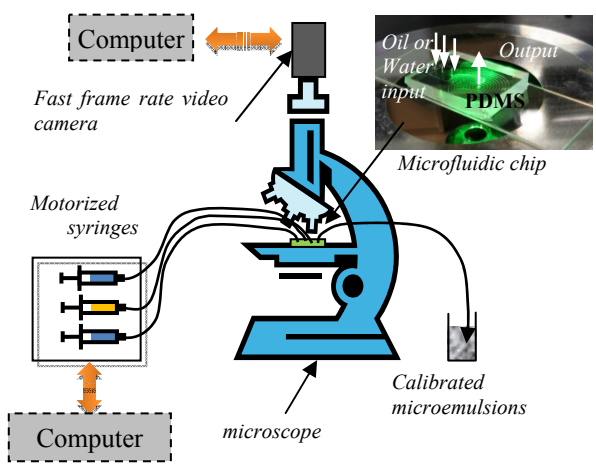


Figure 2: Sketch of the experimental microfluidic system.

CONTROL-COMMAND SYSTEM DESCRIPTION

We use TANGO as the control-command system. TANGO is an object oriented distributed control system based on CORBA and is being actively developed as a collaborative effort between the Alba, Desy, Elettra, ESRF, FRM II, MAX-lab and Soleil institutes.

This system can interface each component of the experience, whether hardware (sensors, motors...) or software, and distribute them over a network. TANGO operates like a software bus, allowing all system components (called device servers) to communicate (see Figure 3).

TANGO provides generic tools for configuration, testing, deployment, supervision. This facilitates the developments and changes to the experiments.

LIONS have developed a package of tools in Python language; enable us to write Python scripts in order to control TANGO device servers in command line.

TANGO Device Servers

The Phantom video camera is interfaced with a C++ device server based on PhCon (control libraries for the Phantom high speed cameras given by Vision Research). For testing, we also use TANGO devices that can publish previously recorded images or video on image attributes.

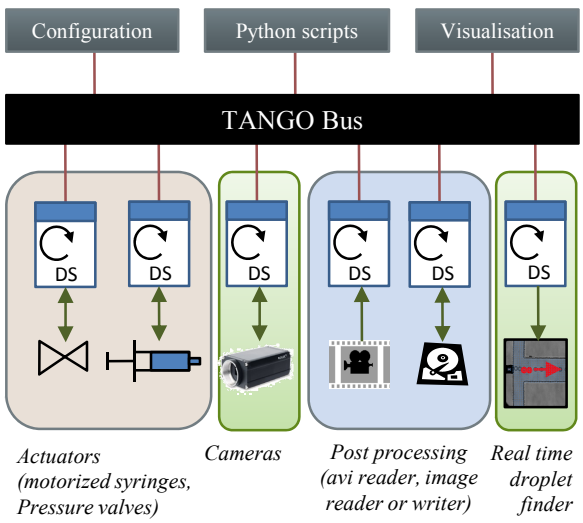


Figure 3: Scheme of the different TANGO layers. Device Servers (DS) are programmed in C++ or Python languages and have the TANGO interface (blue).

Droplet Detection

The "droplets finder" application developed in the present work can connect to all these devices and process the image to find drops. The software is developed in C++, for providing fast results, and is based on the OpenCV [6] (Open Source Computer Vision) libraries, which give programming functions for real time computer vision.

From these libraries, we are using the `cvHoughCircle` function which implements a Hough [7] algorithm and can find circles on a grayscale image. Some parameters have to be specified: maximal radii of circles, minimal distance between circles, and two method (Hough) specific parameters. The function returns a list of circles with their centres coordinates and radii. A post processing (for brightness and contrast) is made on the image for improving the results.

The user can tunes parameters through TANGO. The finding process can be paused in order to test some parameters on the same image. The program yields the averaged radii of found circles and the distance between drops (centres coordinates are sorted vertically or horizontally).

We used Jdraw application (which is a rapid interface designer tool provided with TANGO) for testing the system (see Figure 6), and display the video camera image and the processed image. Circles are drawn over detected droplets.

TIMING CONSIDERATIONS

Image Acquisition

The exposure time on the Phantom camera is typically in between 0.01ms and 0.1ms, and one drop is typically formed in 1ms. This video camera acquires 2000 images/s in full resolution (512x512) and more when decreasing the picture size. For the present application, it is sufficient to have few measurements per second.

We consider that a TANGO device server must respond in less than 300 ms (value widely accepted in the TANGO community and beyond that, a multi-threading system must be programmed). As the acquisition time was very short, we decided to acquire a new image on the camera each time it is asked by a remote TANGO client (when reading the image attribute).

Droplet Detection

The droplet detection device server is compiled with Microsoft Visual C++ 2008 express version, in "debug" mode, and "release" mode, with the corresponding TANGO and OpenCV libraries. Execution times of the droplet detection (see Table 1) are given by the software, and have been obtained on a simple laptop computer with the following configuration: Intel Core 2 Duo P8600 @2.40 GHz and 4 Go ram.

Table 1 : Processing Time for Droplet Detection

Number of detected circles	Detection time	*Complete processing time*
6 (debug mode)	170ms	240ms
6 (release mode)	80ms	120ms
17 (debug mode)	430ms	500ms
17 (released mode)	200ms	240ms

The complete process consists of image access from remote device (on the same computer), a pre-processing of image, search of circles, sort of circles, and a publication of the results over the TANGO device server. This process is separated in his own thread on the device server in order to let the server answer to remote clients.

We can notice that detection time is increasing when there are more circles to find on the image. As we can expect, the software is quicker in "releases" mode of C++ compilation, and processing time is reduce by twice.

Image access to remote servers and all the pre-processing procedure takes only around 70ms in the debug mode, and 40ms in the released mode, which is really acceptable for 512x512 pixels by image. There is no difference of timing when the device server is started as a Windows service by the TANGO starter application.

RESULTS

Drops are produced in the microfluidic chip. We have check that number of drops, sizes and distances given by the software are corresponding to real values by a remote analysis of the images with ImageJ [8], an open source image processing program. The device server can also detect a high number of drops with different radii (from 10µm to 30µm).

By varying rate of water or oil in injected by motorized syringes, we can measure in real time the variation of size (Figure 4), the number of drops (Figure 5) or the distance between drops. During measurements, the parameters for detection are not changed. Values have been obtains by averaging 10 images for each rate.

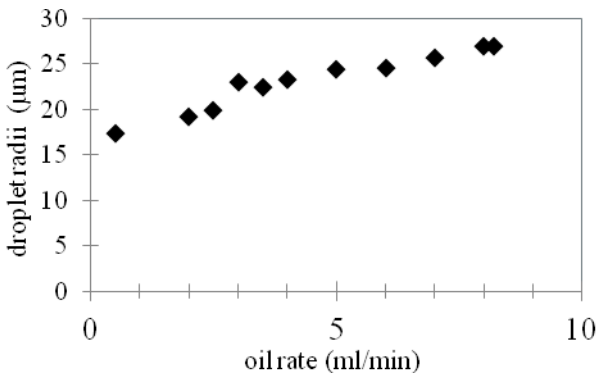


Figure 4: Measured droplet radii obtained by detection when oil rate is varying. The water rate is constant (5µl/min and 80µl/min).

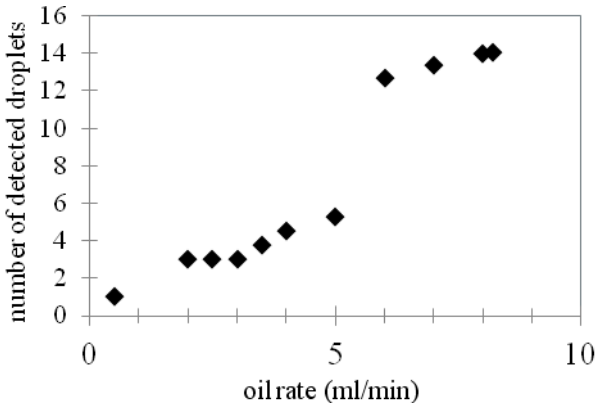


Figure 5: Number of detected droplets when the oil rate is varying. The water rate is constant (5µl/min and 80µl/min). The gap observed is due to real evolution of droplets, and not of detection errors.

Discussion and Future Work

The droplet detection system is working and allows the records of droplets evolution. However for the moment only 80% of droplets are detected. Moreover, the size measured is not always correct. This is the reason why we decided to average the result over 10 images. One possibility to avoid such errors will be to upgrade the software, i.e. by adding a system of averaging on the past images.

Detection process could also be improved: for the moment we try to find circles but; it could be interesting to detect "joined lines", and make a selection by fitting circles or ellipses. OpenCV offer some functions to do that. We could also try to detect "blobs", regions in the image that are either brighter or darker than the surrounding.

CONCLUSION

We have developed an automatic image analysis software with droplet detection for microfluidic systems. Microfluidic is a technic where electronic equipments are important. We demonstrate here that TANGO control command system, normally designed for synchrotron, is usable for more simple laboratory applications.

Our droplet detection is fully functional. Averaged radii and number of droplets are displayed almost in real time. This application can help researchers to control the micro fabrication parameters.

We thank Clélia Timsit, for the experiments preparation.

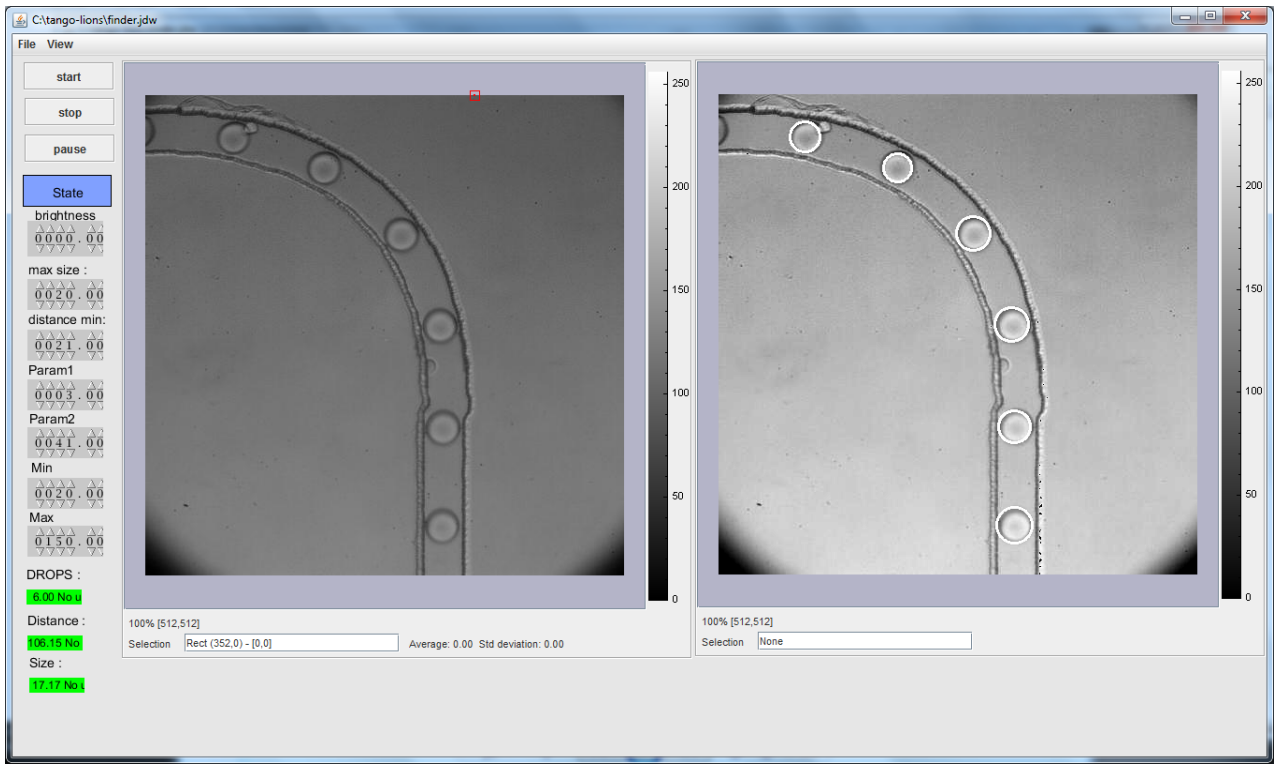


Figure 6: Interface of the application. On the left, the user can start the detection process and tune parameters in real time. Left image is the video camera display. Right image is the processed image, with detected circles drawn on it. Number, size (radii) of drops, and averaged distance between drops are indicated.

REFERENCES

[1] A.Götz and al,"A CORBA-based Control System", ICALEPCS 2003
[2] McDonald, J.C. *et al.* "Fabrication of microfluidic systems in poly(dimethylsiloxane)". *Electrophoresis*, **21**, 27-40 (2000).
[3] Garstecki P, Stone H A and Whitesides G M 2005 *Phys. Rev. Lett.* **94** 164501
[4] http://www.cetoni.de
[5] http://www.visionresearch.com
[6] http://opencv.willowgarage.com/wiki/
[7] Yuen, H. K.and al,"Comparative study of Hough transform methods for circle finding", *Image Vision Comput.*, Butterworth-Heinemann, **8**, 1990, 71-77
[8] http://rsbweb.nih.gov/ij

THE CODAC SOFTWARE DISTRIBUTION FOR THE ITER PLANT SYSTEMS

F. Di Maio, L. Abadie, C. Kim, K. Mahajan, P. Makijarvi, D. Stepanov, N. Utzel, A. Wallander
ITER Organization, Route de Vinon sur Verdon, 13115 Saint Paul Lez Durance, France.

Abstract

Most of the systems that constitutes the ITER plant will be built and supplied by the seven ITER domestic agencies. These plant systems will require their own Instrumentation and Control (I&C) that will be procured by the various suppliers.

For improving the homogeneity of these plant system I&C, the CODAC group, that is in charge of the ITER control system, is promoting standardized solutions at project level and makes available, as a support for these standards, the software for the development and tests of the plant system I&C.

The CODAC Core System is built by the ITER Organization and distributed to all ITER partners. It includes the ITER standard operating system, RHEL, and the ITER standard control framework, EPICS, as well as some ITER specific tools, mostly for configuration management, and ITER specific software modules, such as drivers for standard I/O boards.

A process for the distribution and support is in place since the first release, in February 2010, and has been continuously improved to support the development and distribution of the following versions[1].

THE CODAC CORE SYSTEM DISTRIBUTION

Plant System I&C Components

The architecture of the Instrumentation and Control (I&C) for the ITER plant systems is illustrated in Figure 1.

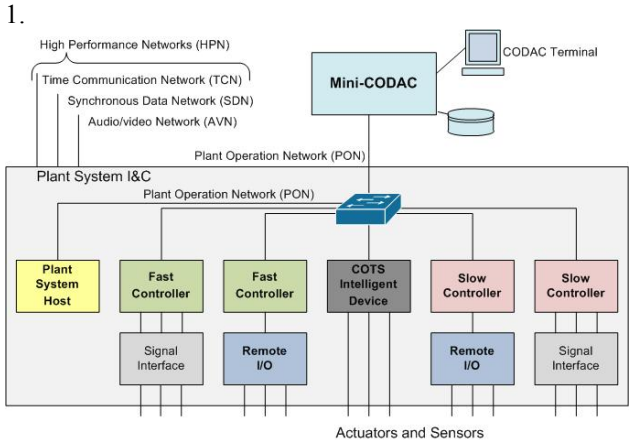


Figure 1: I&C Architecture before integration.

The components are:

- Siemens PLC for "slow" conventional control (up to 10 ms loops) and for "slow" interlock controls (up to 100 ms loops).
- Fast controllers, which currently are PICMG compliant industrial computers controlling a PXI/PXIe/cPCI remote I/O chassis over a PCIe link. The operating system is Red Hat Enterprise Linux (RHEL) that can be enhanced with real-time extensions for applications requiring high predictability.
- A system named Plant System Host (PSH) to implement the standard services, such as operating state management or health monitoring. The PSH is similar to a fast controller except that it includes no specific I/O. The PSH is also a gateway to communicate with the Siemens PLCs.

Fast interlock controllers as well as high range xTCA I/O for fast controllers are under development and will be within scope of future distributions.

Mini-CODAC

The CODAC acronym means "Control Data Access and Communications" and also designates the central control infrastructure at ITER. During the development phase of a Plant System, the central CODAC services are replaced by a Mini-CODAC system (see Fig. 1). It is both a system development and a runtime platform while providing a subset of CODAC central services, such as alarm handling and archiving facilities.

The operator software on Mini-CODAC is built with Control System Studio (CSS).

The Software Distribution Variants

The CODAC Core System distribution includes the software required for the Mini-CODAC, the fast controllers, the PSH and for additional operator consoles. The distribution covers both the development of the plant system I&C and its operation, during tests. Two set of packages are made available: "development" and "production". The packages that will be installed on a particular system are determined according to the role that the user assigns to the system when installing the CODAC Core System distribution.

For each platform, the distribution bundles, as a common base, the operating system and the EPICS framework.

PLC Support

The development for PLC is based on the Siemens Step-7 software and is not within scope of the CODAC Core System distribution. The CODAC Core System includes however the standard templates for Siemens Step-7 software that shall be used for developing the PLC software and the communication software for the

integration of the PLCs into the ITER controls are included.

The development software for Mini-CODAC and PSH includes the tools to declare variables on PLCs and to generate the communication software to exchange data between EPICS variables and the PLC ones. The generated variable declarations shall be included in the PLC application software by the developer.

Fast Controllers Support

The ITER Organization issues some hardware catalogues for the supported I&C components covering the recommended hardware modules for the PLC and the fast controllers.

A set of PXI/PXIe boards is defined in the fast controller catalogue, which is regularly extended. The software for supporting these selected boards will be added in the distribution in an incremental manner

The current distribution includes the software for the following boards:

- NI PXI-6259: a multi-function data acquisition board (16 analog input, 4 analog output, 32 digital I/O).
- NI PXI-6682: a synchronization and timing board (IEEE-1588)

The software for the following boards is under development and will be included in the 2012 versions:

- NI PXIe-6368: a higher performance multi-function data acquisition board (16 simultaneous analog inputs at 2 MS/s)
- NI PXI-6528: a digital I/O board (24 input, 24 output, optically isolated)
- NI FlexRIO boards (FPGA) with adapter boards. In particular PXI-7951R with 6581 (digital) and 5781 (analog) adapters. The software shall be compliant with other FlexRIO hardware and also with cRIO I/O boards connected over PCIe interconnect link.

For each board, the software support includes the Linux device driver and the EPICS device support.

The configuration tools (SDD, see below) are also extended to allow generating the configuration files for these boards from the controller's configuration.

CODAC Core System Versions

The CODAC Core system distribution is produced with a major release every year, very few (1-2) intermediate minor in between and maintenance releases for distributing bug fixes or minor extensions whenever required[2].

The CODAC Core System versions are defined as: *major_id.minor_id.maintenance_id*. Examples: 3.0.0 is the base release for the version 3 and 2.1.1 is the 1st maintenance release for the 2.1 version.

The installation of a maintenance release will update an installed version so, by default, the application software does not require being re-built. A minor release shall be backward compatible with another minor release within the same major version so, in principle, the application software requires being re-built but not being modified. Major release may include a new version of the operating system or of EPICS and, up to now, require a full installation.

Preliminary versions are also produced before any official release with alpha/beta identifiers replacing the maintenance release number (ex: 3.0b1 for the 1st beta version of 3.0).

Self-Description Data (SDD) Tools

The promoted approach for ITER controls is to have the configuration data for the plant system I&C structured with a common schema defined at project level [3].

This is implemented in the CODAC Core System with a set of configuration management tools that maintain the plant system I&C definition in relational databases

This definition includes:

- The functions
- The variables (EPICS PVs, PLC variables)
- The signals
- The controllers with their I/O boards
- The links between variables, signals and I/O boards
- The configuration for alarms
- The configuration for archiving

When the CODAC core system distribution is installed on a Mini-CODAC system, a local database is configured in order to store/retrieve the plant system I&C definitions. In IO, a central repository is also maintained. The local databases are initialized with data copied from the central database and can be synchronized with the central ones in order to update local copies with the central one or to commit new definitions (see [4] for details).

An editor (SDD editor) allows the creation/edition of the plant system I&C into the local database. A web application (SDD browser) is also provided to allow accessing the plant system I&C definitions from web browsers.

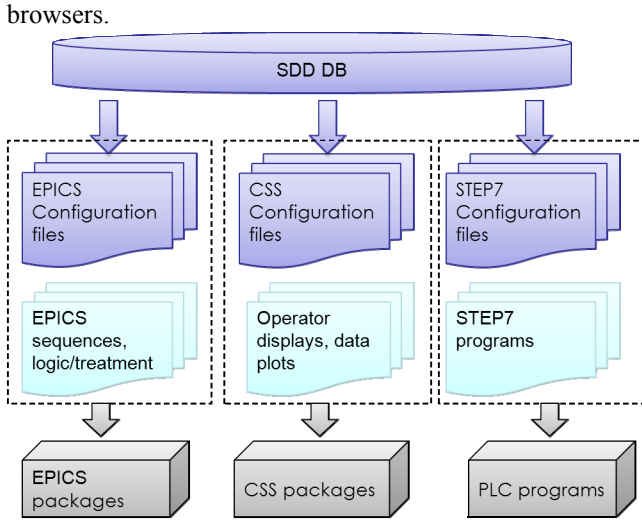


Figure 2: SDD generated configuration files.

As illustrated in Figure 2, the SDD toolkit includes translators for generating all specific files:

- the build files (makefile, scripts…),
- the EPICS configuration files (record databases, IOC scripts…),

– the Control System Studio (CSS) configuration files for alarms and archiving,
– the variables declaration for the PLCs.

The EPICS database files, which define the set of EPICS records implementing the application, are usually edited with text editor or graphical "database configuration tool" such as VDCT. In the ITER environment, these files are generated by the SDD tools but, it will also be possible, in the 2012 version, to use EPICS tools to edit these files and have any resulting change or addition retrofitted into the SDD database.

The SDD tools also include validation functions, which allow verifying the completeness of the definition during the development and the compliance with ITER standards, such as the naming convention, before delivery.

BUILD OF THE CODAC CORE SYSTEM DISTRIBUTION

Different software distributions are built for the different "system roles": Mini-CODAC/development, Mni-CODAC/Operation, PSH/Operation, etc.

These distributions are built from:

1. The operating system.
   RHEL 6.1 will be provided with the CODAC Core System vs3 in 2012 (today: 5.5). RHEL-MRG-R (6.1 with real-time extensions) can also be provided with the fast controller distributions.
2. EPICS components.
   It includes the EPICS base and EPICS modules, such as the SNL sequencer, CAJ/JCA (Java client library) and VDCT.
3. Control System Studio (CSS) and CSS applications: BOY, BEAST, BEAUTY and SNL editor, in the Mini-CODAC distributions.
4. ITER specific development tools.
   Self-description data (SDD) tools and maven-based build process are included in the "development" distributions.
5. Software support for the standard I/O modules in the fast controller distributions.

Changes made by ITER in the EPICS components, such as extensions for 64 bits or support for the PostgreSQL database are, as far as possible, retrofitted in the shared EPICS sources but the ITER distributions are only built from the ITER source repository (SVN).

ITER Build Tools

The steps for building software components from the source code in repository to the packages in installation servers are supported by commands implemented with the Apache maven (mvn) tools.

The system packages as well as the application ones are built with standardized mvn commands:

- mvn checkout - extract a software unit from the source repository (using versions' branch or tag)
- mvn compile – build all object files
- mvn test – run the tests of the unit
- mvn package – build the RPMs for deploying the unit to target hosts.

These commands are executed by the developers during development and, when a unit is included in the distribution, by the build servers for producing the different software releases. Build servers are configured with the Jenkins continuous integration tool.

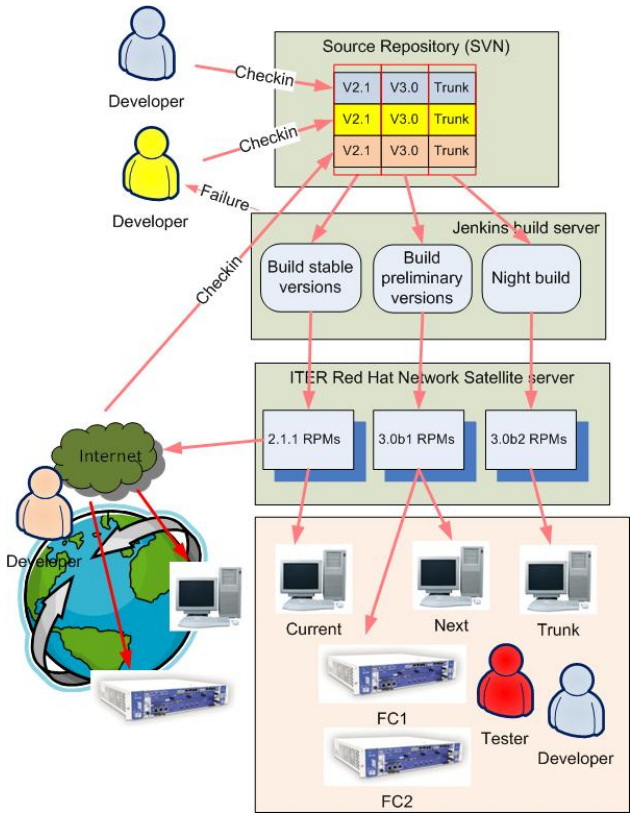


Figure 3: Build and distribution process.

The next actions executed by the build servers are the following:

- Sign the RPMS and copy them to network shared directories.
- Copy the official releases RPM, tagged with a version id, to the distribution servers at release time.
- Copy the RPMs that are generated for continuous integration, tagged with a branch id and a SVN id, to distribution servers every night.

Test computers are thereafter updated with new releases, automatically for the nightly built ones.

Whenever a compilation or a test executed by a build server fails, the package owner is informed by mail. Build logs can be consulted with the Jenkins tools.

Verification of the Java code with static code analysis tools, such as CheckStyle, will be added in order to also provide the developers with reports on the quality weaknesses detected in their code.

Installation of the ITER Distribution

The ITER distribution servers are configured with the Red Hat Network Satellite management system. The

operating system distribution, the ITER packages distribution and the subscription management are centralized on these servers.

Users are grouped into organizations that can be remotely administrated by authorized users for allocating RHEL subscriptions and configuring the local systems.

For any installation, the user shall have access to the distribution servers. However, once configured, the local systems do not require permanent access to the ITER servers, except for delivering software in the ITER repositories or for accessing support information.

Installing a local system is an automated procedure using the IO supplied installation disk images. They provide a selection of the CODAC Core System, bundled with the RHEL operating system and a suitable role for the target system.

CONCLUSION

The integrated process for building and distributing the CODAC software allows automating the compilation, testing and packaging of software components from different sources. It also allows maintaining a unique distribution system for the systems in Cadarache and the ones used by the suppliers and partners that are very widely distributed.

It requires constant resources for the maintenance but the automated processes reduce a lot the time and cost for the integration and tests of the software releases. This is very valuable to cope with the increase of the test and maintenance releases that will be required during the development phase of the plant system I&C.

REFERENCES

[1] A. Wallander et al., “News from ITER Controls – A Status Report”, this conference.

[2] A. Zagar et al., “ITER CODAC Core System Development Process”, IAEA TM8, San Francisco, June 2011

[3] L. Abadie et al., “The self-description data configuration model”, IAEA TM8, San Francisco, June 2011.

[4] D. Stepanov et al. “Integrated approach to the development of the ITER control system configuration data”, this conference.

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization

BEAM SHARING BETWEEN THE THERAPY AND A SECONDARY USER

K. Gajewski, The Svedberg Laboratory, Uppsala University, Uppsala, Sweden

Abstract

The 180 MeV proton beam from the cyclotron at The Svedberg Laboratory is primarily used for patient treatment. Because of the fact that the proton beam is needed only during a small fraction of time scheduled for the treatment, there is a possibility to divert the beam to another location to be used by a secondary user. The therapy personnel (primary user) control the beam switching process after an initial set-up which is done by the cyclotron operator. They have an interface that allows controlling the accelerator and the beam line in all aspects needed for performing the treatment. The cyclotron operator is involved only if any problem occurs. The secondary user has its own interface that allows a limited access to the accelerators control system. Using this interface it is possible to start and stop the beam when it is not used for the therapy, grant access to the experimental hall and monitor the beam properties. The tools and procedures for the beam sharing between the primary and the secondary user are presented in the paper.

INTRODUCTION

The synchrocyclotron [1] at The Svedberg Laboratory (TSL), originally built in the early fifties and almost totally reconstructed in the years 1977-86, can accelerate protons and different ions, ranging from helium to xenon. Until 2005 TSL had been so called national facility and was financed from the state budget. The research groups, both from Sweden and abroad, could send the proposals for an experiment to be done at TSL. The beam time has been granted to the projects selected by the lab's Program Advisory Committee. After closing down CELSIUS storage ring at TSL in 2005, the laboratory has been reorganized and changed the orientation from being the laboratory hosting the basic research experiments to an accelerator facility dedicated mainly to patients' irradiations with protons. TSL is financed now by the Uppsala County Council (the patient treatment) and by selling the beam time to the other clients. The contract signed between TSL and the University Hospital in Uppsala specifies that thirty five weeks a year are reserved for the patients' treatment. Because the proton beam is needed in the treatment room only for a small fraction of time scheduled for the therapy irradiation, there is a possibility to use the beam in other experimental areas of the laboratory at "the same" time.

BEAM SHARING PREREQUISITES

The beam is shared between the irradiation therapy (primary user) and users in other experimental locations (secondary users). The switching between two different beam lines is done with help of a bending magnet but a number of other parameters are also changed in order to satisfy the beam requirements of both therapy and the secondary user. The beam that is used for the therapy irradiation is a 180 MeV proton beam and although the beam characteristics can be different for the therapy and the secondary user, we can't change neither particle nor it's energy when the beam is shared.

During the "therapy" weeks the beam is available for patients' irradiation according to the following schedule:

- Monday 1 pm. - 8 p.m.
- Tuesday-Friday 8 a.m. – 6 p.m.

The number of patients is usually between 6 and 9 per day and each patient receives two or three fractions (irradiations from different angles). The beam time needed for each fraction is usually less than one minute and repositioning of the patient between the fractions takes about 5 minutes. The quality assurance tests performed every morning before the first treatment require the beam for about an hour. Thus the real beam time (adding usual delays and a beam switching time) using by the therapy is about 3 hours (counting 5 min. for one fraction) a day.

THE PRIMARY USER

The primary user is the one that controls where the beam is led. He can take away the beam from the secondary user at any moment. There is a delay (which is set by agreement between the therapy and a secondary user) between the request to take away the beam from the secondary user and actually doing it. After that delay the beam is switched to the primary user. When taking the beam from the secondary user the therapy operator specifies the expected time (based on number of fractions) the beam will be not available for the secondary user. When sending the beam back to the secondary user, the therapy operator specifies also the approximate time the beam will be available for him before the next switching. The control of the dose delivered to the patient is the responsibility of the therapy operator and is independent of the accelerator control system.

THE SECONDARY USER

The secondary user gets the beam when the primary user decides to release it. The exact moment when it happens is unknown to the secondary user but there is information when the beam is supposed to be switched. The same is true for taking away the beam from the secondary user, with the exception that there is a delay between the beam request from the therapy and actual switching the beam. This delay is set during the beam switching set-up and depends on the secondary user's requirements.

The secondary user has a very limited access to the control system. He can perform the following actions:

- Forbid and allow access to his experimental hall (clear / release clearing). The hall must be in "cleared" state to put the beam into it.
- Monitor the parameters important for the experiment
- Stop and start the beam if it is available for him or enable/disable the beam when it is used by the therapy. When the beam is directed to the secondary user and it is enabled, it will be turned on, otherwise it will be turned off and the secondary user can start the beam at any time as long the beam is available for him.

BEAM SHARING SETUP

Before the therapy operator can use the switching procedure the cyclotron operator must set it up. The following steps must be taken:

- Decide what parameters should be changed during the switching process in order to fulfill the requirements for the beam properties (such as intensity, focus, position on target) both for the therapy and the secondary user
- Set the required delay between the beam switching request and actual beam switching when taking the beam from the secondary user
- Set the beam current limits for both therapy and the secondary user
- Set-up the required beam properties to both therapy and the secondary user
- Test the switching procedure
- Set-up the alarm conditions for the secondary user
- Set-up the parameter reservation
- Start/enable the billing for the secondary user's beam time

When the procedures listed above are completed, the beam switching can be performed by the therapy operator without any help from the cyclotron operator. In fact, the cyclotron operator is usually not longer present in the control room but is doing other job in the laboratory and is available "on call" in case of unexpected problems.

SERVICES

There are a number of programs that are used to make it possible to share the beam between the therapy and the secondary user without the need for the cyclotron operator. All these programs use the services that are essential for such a beam sharing scheme working satisfactory in real life. The most important services are shortly described below.

Logging

All programs used by the therapy operator and the secondary user log all their actions using syslog facility. The programs can (and usually are) executed on different computers but syslog daemons are configured so that all the log messages are saved on one syslog server. The entries in the log file are tagged with the time stamp (all computers are synchronized with the network time server), computer name, user name, log entry priority and the program name. The log file is saved on a week basis and archived on Sunday night (done by a cron job). The log file is an invaluable help when it comes to understand the problem that could have occurred, to spot the unexpected actions of the therapy operator or the secondary user or just to follow the use of the beam. The log file is also used for calculation of the beam time that the secondary user is charged for (billing).

Alarms

As in any system things might go not as planned. The quick observation of the problem and passing information about it to the cyclotron operator is essential. It is also important that some actions will be performed automatically (like stopping the beam and the billing). The alarms from the programs involved in the beam sharing are distributed by two independent systems:

- Local text paging infrastructure (radio transmitter using a license free 439 MHz frequency band, alphanumeric pagers)
- SMS (via email gateway) to the mobile phones. The message is usually delivered within 1 minute (not guaranteed).

The message sent to the pager or the phone contains the time stamp of the event, originating program's name and a short description of the failure.

Parameter Reservation

TSL's control system has a global mechanism for limiting the rights to access control system's parameters based on the user's account uid and gid (uid – user id, gid – group id). The write access rights for every parameter in the control system can be set to the following states based on uid and gid:

- Free – write access for everybody
- Reserved – write access only for the specific uid/gid
- Locked – no write access

After the beam switching is set-up, the cyclotron operator locks the parameters that should not be changed by any user and reserves those that must be changed in the beam switching process for use by the therapy operator. The switching program changes access rights to some parameters during the switching process to allow beam control by the secondary user when the beam is used by them, and to prevent them to control the beam when the beam is used for the therapy.

Billing

The secondary user is charged for the beam time he has to his disposal. It's not possible to exactly predict how much time beam time will be used by the therapy and manually keeping track of it is out of question. Therefore there is a virtual parameter BILLING that can be in the following states:

- **Enabled** - the beam is used by the therapy but as soon as it will be switched to the secondary user and the beam's parameters will be accepted, the state will changed to "Started".

- **Started** - the beam is available to the secondary user, the state will change to "Enabled" when the beam is taken back to the therapy or to "Stopped" in case of some failure.
- **Stopped** - the beam is not available to the secondary user, either because of some failure or deliberate action taken by the cyclotron operator.

The secondary user can see the actual state of the billing in his user interface. The cyclotron operator can at any time generate a billing report.

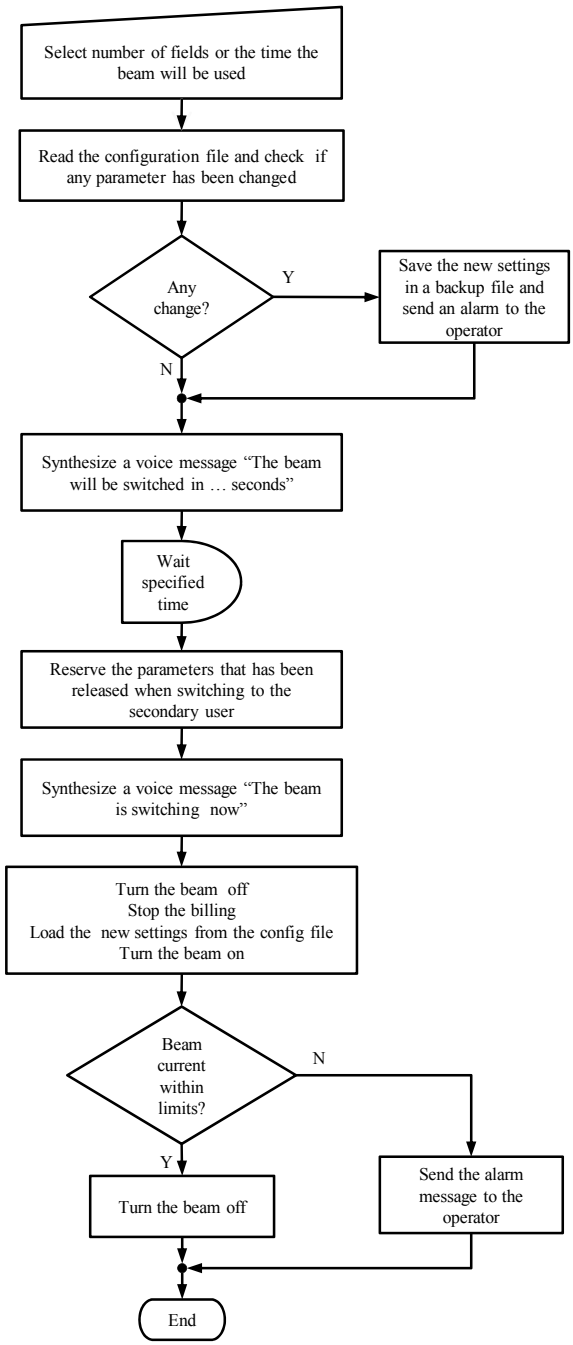


Figure 1: Beam switching from the secondary user to the therapy.

Synthesized Voice Messages

It turned out to be very useful to create the voice notifications during the switching process and distribute them over the network to different locations in the laboratory. Festival speech synthesis system has been used for implementation of this feature.

BEAM SWITCHING

As said before the beam switching process is controlled entirely by the therapy operator. The program to control the beam switching is written in tcl/tk and it has slightly different user interface depending on if it is used by the cyclotron or therapy operator (features needed for the beam switching set-up are not visible in the therapy interface).

The main window has only two buttons and one status field indicating into which location (therapy room or experimental hall) the beam is directed. The buttons are for selecting the area the beam should be switched into.

When one of the buttons is pressed the pop-up dialog and status windows are shown during the switching process.

Figures 1 and 3 show the algorithm implemented for switching the beam from the secondary user to the therapy and vice versa.

USER INTERFACES

The user interfaces are built using TSL's control system standard tools and components. The therapy operator uses a workstation connected directly to the control system network and controls the accelerator via a "synoptic" program.

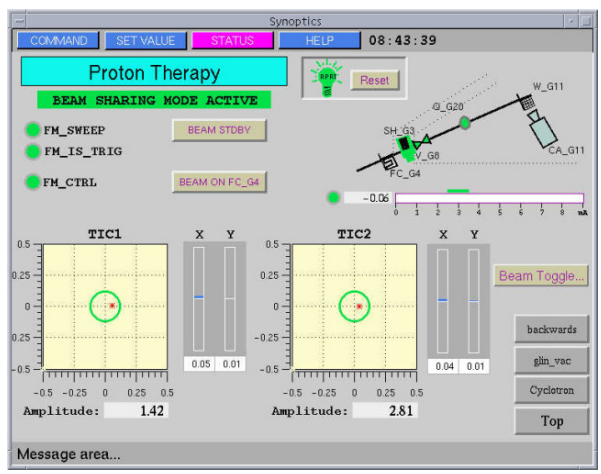


Figure 2: Synoptic screen for the therapy user.

The secondary user works on a workstation connected to the laboratory "public" network and connects to the control system via a firewall using secure shell protocol. The account that he logs into the control system allows only limited access to the control system. The examples of the "synoptic" screens used by the therapy user and the secondary user are shown in figures 2 and 4 respectively.

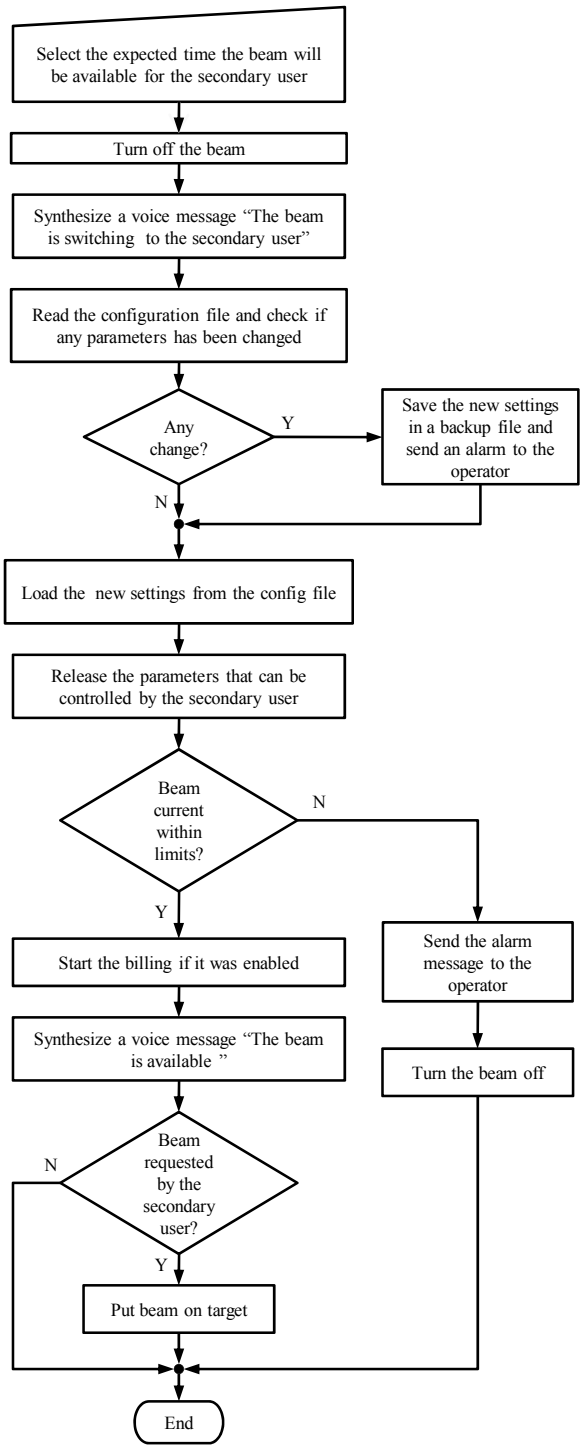


Figure 3: Beam switching from the therapy to the secondary user.

Here follows a short list of the tasks that therapy operator and secondary user can perform using their interfaces to the control system.

Therapy User

- Perform the beam switching process
- Monitor the beam position during the irradiation
- Turn on and off the beam
- Check the beam current before starting the irradiation
- Monitor the parameters in the end of the therapy beam line

Secondary User

- Control the access rights to the experimental hall (radiation protection).
- Request the beam when it is used by the therapy.
- Turn on/off the beam when it is not used by therapy.
- Monitor the parameters relevant for the specific experiment (beam current, signals from particle detectors, position of the target, billing status)

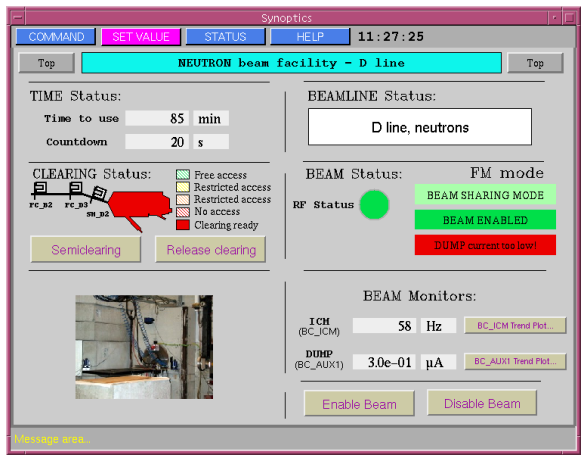


Figure 4: Synoptic screen for the secondary user.

CONCLUSIONS

The beam sharing described in this paper has routinely been used at TSL for the last few years. During this time many adjustments has been done to improve the quality of the system and to make the switch process easy to use for the therapy operator. Several synoptic screens for different beam lines and test stations have been created for the secondary users. A possibility to control the beam current by the secondary user is planned to be implemented in the near future.

REFERENCES

[1] "TSL Progress Report 1987-1991", The Svedberg Laboratory, Uppsala, Sweden, p. 10

INTEGRATION OF THE MOMENT-BASED BEAM-DYNAMICS SIMULATION TOOL V-CODE INTO THE S-DALINAC CONTROL SYSTEM*

Sylvain Franke**, Wolfgang Ackermann, Thomas Weiland,
TEMF, Technische Universität Darmstadt, Darmstadt, Germany

Ralf Eichhorn, Florian Hug, Christian Klose, Norbert Pietralla, Markus Platz
IKP, Technische Universität Darmstadt, Darmstadt, Germany

Abstract

Fast and accurate beam-dynamics simulation programs within accelerator control systems can advantageously assist the operators and provide a more detailed insight into the actual machine status. The V-Code simulation tool implemented at TEMF is a fast tracking code based on the moment approach. In this contribution an overview of the numerical model is presented together with implemented features for its dedicated integration into the control system of the superconducting linear accelerator S-DALINAC.

INTRODUCTION

The Superconducting Darmstadt Linear Accelerator S-DALINAC [1], installed at the institute for nuclear physics (IKP) at the Technische Universität Darmstadt is designed to allow nuclear- and astrophysical experiments with electron beams in an energy range from 1 MeV up to 130 MeV with beam currents up to 60 μA. The electrons are emitted from a 250 keV thermionic gun or alternatively from a recently installed source for 100 keV spin-polarized electrons.

The beam is bunched in the chopper and prebuncher section in order to prepare the particles for the acceleration within the radio frequency cavities. The provided bunches are first accelerated up to 10 MeV within the injector linac and subsequently deflected by 180 degrees into the main linac. This main linac section is composed of 8 superconducting 20-cell cavities and is designed to provide an energy gain up to 40 MeV. The main linac can be passed up to three times using the beamlines of the first and second recirculation, leading to a maximum bunch energy of 130 MeV.

The layout for beam recirculation shown in Fig. 1 together with the few beam diagnostic instruments render it quite complicated to find optimum settings for the individual beamline elements. Fast online beam-dynamics simulations can assist the operators because they provide a more detailed insight into the actual machine status.

The moment approach allows to implement a beam-dynamics simulation tool that is on the one hand much faster than classical beam-dynamics simulation codes which track numerous macro particles and on the other hand more accurate than matrix-based beam-dynamics codes.

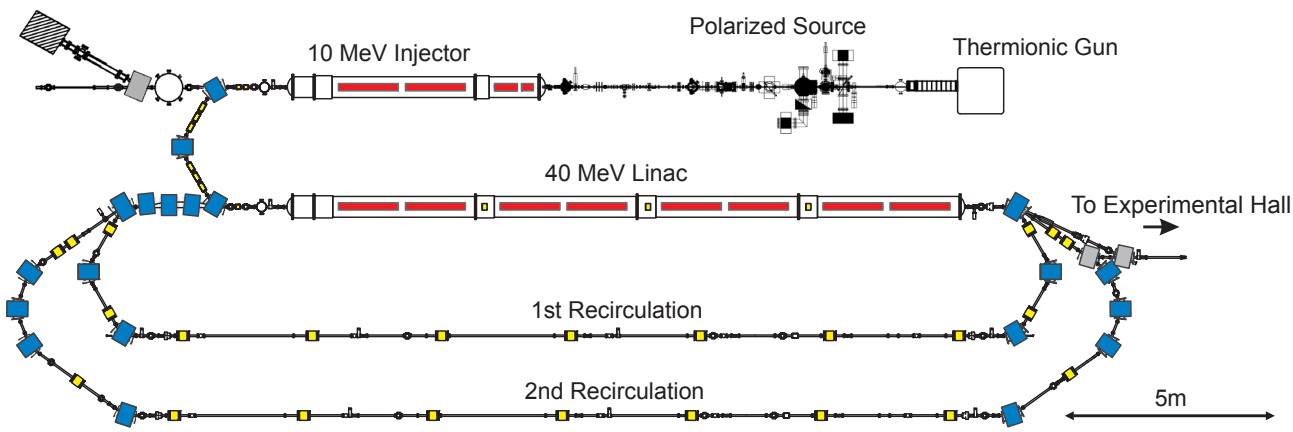


Figure 1: Schematic layout of the Superconducting Darmstadt Linear Accelerator S-DALINAC.

* Work supported by DFG through SFB 634.
** franke@temf.tu-darmstadt.de

THE MOMENT APPROACH

The moment approach to beam-dynamics consists in representing the particle distribution function f in the 6-dimensional phase space Ω spanned by the Cartesian coordinates x, y, z and the momentums p_x, p_y, p_z by a discrete set of characteristic moments [2].

Moment Definition

The moments are obtained by a weighted integration over the whole phase space

$$<\mu> = \int_{\Omega} \mu \; f(\vec{r}, \vec{p}, \tau) \; \mathrm{d}\Omega. \qquad (1)$$

A numerically advantageous choice is given by the first order raw moments $\mu \in \{x, y, z, p_x, p_y, p_z\}$ and centralized moments

$$\mu \in \{(x - <x>)^{l_1} \cdot \; \ldots \; \cdot (p_z - <p_z>)^{l_6}, \; \ldots\} \; ,$$

for the higher orders $n = l_1 + ... + l_6 \in \mathbb{N}$, $n \geq 2$.

Following this idea one automatically obtains the center of mass position and a translatory invariant description of the shape of the particle distribution [3].

Moment Evolution

A fundamental time evolution equation can be stated for each moment $<\mu>$ by means of the Vlasov equation:

$$\frac{\partial <\mu>}{\partial \tau} = <\frac{\partial \mu}{\partial <\vec{r}>}><\frac{\vec{p}}{\gamma}> + <\frac{\partial \mu}{\partial <\vec{p}>}><\frac{\vec{F}}{m_0 c^2}> + <\frac{\partial \mu}{\partial \vec{r}} \frac{\vec{p}}{\gamma}> + <\frac{\partial \mu}{\partial \vec{p}} \frac{\vec{F}}{m_0 c^2}> \qquad (2)$$

This set of ordinary differential equations can efficiently be evaluated with standard time integration methods [4] if proper initial conditions are given and all essential forces are known.

Force Calculation

The Lorentz force $\vec{F} = q \cdot (\vec{E} + \vec{v} \times \vec{B})$ specified in the fundamental time evolution equation (2) can be evaluated as a sum of forces resulting from superposed external fields and internal space charge effects. A numerically useful way to provide all external field components in the vicinity of the particle distribution is given by a paraxial approximation using a series expansion of the external fields. For the calculation of internal space charge forces a model assuming a certain particle distribution can be applied to reconstruct the distribution within the bunch from the moment description. An alternative approach consists of representing the particle distribution through multiple interacting sets of moments in a multi-ensemble environment [5].

IMPLEMENTATION

The beam-dynamics simulation tool V-Code has been implemented at the Computational Electromagnetics Laboratory (TEMF) at the Technische Universität Darmstadt on the basis of the moment approach.

Beamline Representation

For applying the moment approach even to a long accelerator beamline it is useful, especially in terms of an efficient memory management, to represent the beamline as a consecutive chain of several independent beamline elements along a straight axis. Drift spaces between beam forming elements are represented by beamline elements which are free of external electromagnetic fields. Within bending magnets a reference path has to be defined. This path fixes the length of the magnet along the projected axis within V-Code. The actual bunch position within the bending magnets is then determined via a normal projection onto the reference path [6].

Recirculations

Several beamline elements are repassed multiple times by the beam during the two recirculations. Beamline elements that are passed n times are placed n times along the beamline representation within the V-Code. Proper reference paths have to be defined for all instances of a bending magnet that is passed by the beam at different energies during the recirculations.

The length of the S-DALINAC beamline can be adjusted mechanically in order to match the path length of the recirculated beam with the phase of the fields inside the cavities. This is realized by means of extendable bends within the first and the second recirculation. In V-Code this path length variation is implemented through drift spaces with a variable length placed in front of and behind the movable beamline elements.

Overlapping Fields

The proximity of some beamline elements leads to overlapping fringe fields between consecutive elements. The V-Code database of disjunctive beamline elements has to be extended accordingly.

The fundamental time evolution equation (2) can be separated into drift terms and transport terms. The drift terms represent the fundamental internal feedback and depend on the momentum whereas the transport terms consider the interaction with the external forces. If linear materials can be assumed, the specified force can be evaluated as a sum of forces resulting from various superposed fields and it is also allowed to apply one transport term for each overlapping field component [7]. This approach was implemented in a new enveloping macro element which enables the user to combine all beamline elements with overlapping fields.

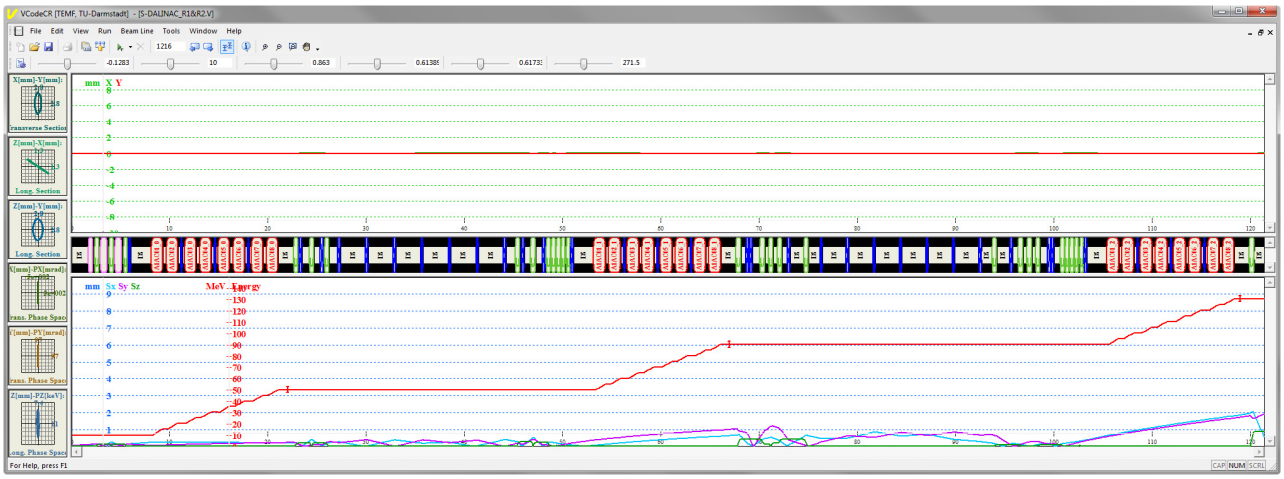


Figure 2: Screenshot of the V-Code graphical user interface showing simulation results for the S-DALINAC recirculating machine. The simulation was initiated right after the injector and the corresponding 10 MeV bunch was subsequently accelerated three times within the main linac. Reference markers for 50, 90 and 130 MeV beam energy are displayed.

User Interface

A comfortable and clearly arranged graphical user interface is essential in order to provide the operators of the accelerator a utile simulation tool. Figure 2 shows a screenshot of the V-Code user interface. Several beam parameters like the position of the center of mass in x and y direction, the dimensions of the beam σ_x, σ_y and σ_z in Cartesian coordinates and the energy are displayed along the longitudinal axis z in the panes above and below the representation of the beamline. On the left side six phase space projections on different planes can be observed at any position in time.

The user interface also provides access to each parameter of the various beamline elements by double clicking on a specific element in the beamline representation. Another way to tune the settings of the beamline elements is implemented via arbitrary assignable sliders in the menu bar.

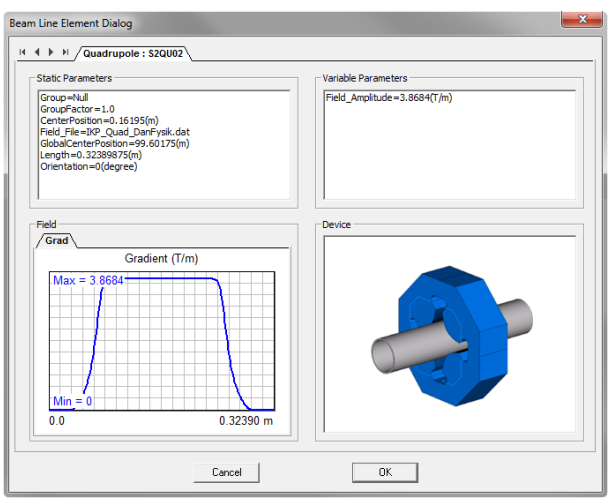


Figure 3: Dialog providing access to all static and variable parameters of a quadrupole. In addition the distribution of the quadrupole strength and a schematic representation of the magnet are displayed.

Online Simulation

Supplementary to the possibility to manipulate the parameter setup of the beamline elements within the V-Code through the user interface an online simulation mode has been implemented. Within this operating mode the parameter settings are read from an external source. This external source may be either an other software tool or for example the control desk of the accelerator machine.

Markers

During each simulation run the actual bunch parameters at the interfaces between the consecutive beamline elements can be stored. These specific sets of moments can be used as initial start ensembles for upcoming simulations of subsets of the beamline. This feature helps to further speed up the iterative process of finding an optimal machine setup.

Reference Markers

In order to additionally assist the operator, reference markers can be displayed on top of the simulation data. This can be used for instance to superimpose expected data or measurement data received from a diagnostic device installed within the beamline.

Automatic Beam Adjustment

Finding an optimal parameter setup for the S-DALINAC beamline is a laborious task as the number of variable parameters is very large. The V-Code opens up the possibility to analyze a multitude of parameter sets in a reasonable time. This procedure can be automated by implementing objectives and evaluation rules. Further, a connection between the V-Code and the accelerator diagnostic software allows to take into account measured parameters in the automated optimization process. A promising prototype for an automatic beam adjustment algorithm applying the V-Code has been published in [8].

APPLICATION

The V-Code has to be accurately initialized with the geometrical setup of the beamline, the field distribution within the various beamline elements and an initial set of moments for its dedicated application within the S-DALINAC control system.

Initialization

The S-DALINAC beamline shown in Fig. 1 consists of over 100 beam guiding elements and for each of them the electromagnetic fields have to be provided. The field distribution along the reference paths of the dipoles, the quadrupole strength along the axis within the quadrupoles and the accelerating electric field within the cavities were extracted from field calculations with the appropriate solvers included in the CST STUDIO SUITE [9].

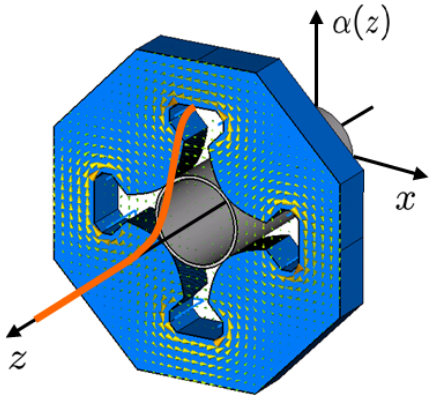


Figure 4: Computational model of a quadrupole magnet together with the extracted quadrupole strength $\alpha(z)$.

The position of the beamline elements within the V-Code beamline is defined by the length of the field data sets of the consecutive beam guiding elements and drift space elements which represent intervals without external electromagnetic fields.

Within the S-DALINAC beamline no diagnostic device exists that is able to measure at one position all the bunch parameters required to initialize the V-Code properly. Nevertheless, the measured bunch data can be used to ensure the bunch parameters achieved from start-to-end simulations.

Simulation Results

The V-Code was initialized with values provided by experienced operators of the S-DALINAC in order to examine its applicability to simulate the recirculating setup. The simulation was started right after the injector section with a 10 MeV bunch with the dimensions $\sigma_x = 0.060$ mm, $\sigma_y = 0.060$ mm, $\sigma_z = 0.014$ mm and a transversal bunch divergence of 0.1 mrad. Results of this simulation are displayed in the screenshot in Fig. 2. More details concerning the simulation of the S-DALINAC recirculations are published in [10].

SUMMARY

The moment approach together with a paraxial field approximation allows the implementation of a fast and accurate beam-dynamics simulation tool. Changing the order of moments or the number of applied ensembles permits to adapt the simulation to the required accuracy while keeping the simulation times in an acceptable level. With the help of the reference-path concept this code can be used to simulate even recirculating machines like the S-DALINAC. With its customized graphical user interface the V-Code is designed to assist the operators in finding an optimal setup for the accelerator beamline.

REFERENCES

[1] R. Eichhorn: Optimierung des Strahltransportsystems und experimentelle Umsetzung verschiedener Methoden zur Gütemessung am S-DALINAC, Dissertation, Darmstadt, Germany, 1999.

[2] P.J. Channell: The Moment Approach to Charged Particle Beam Dynamics, *IEEE Trans. Nucl. Sci.* , NS-30, No. 4, Page 2607, 1983.

[3] M. Krassilnikov, A. Novokhatski, B. Schillinger, S. Setzer, T. Weiland, W. Koch: V-Code Beam Dynamics Simulation , Proceedings of the 6th International Computational Accelerator Physics Conference (ICAP 2000), September 2000.

[4] W. Ackermann, T. Weiland: Efficient Time Integration for Beam Dynamics Simulations Based on the Moment Method, Proceedings of ICAP 2006, Chamonix Mont-Blanc, France, October 2006.

[5] S. Franke, W. Ackermann, T. Weiland: A Fast and Universal Vlasov Solver for Beam Dynamics Simulations in 3D. Proceedings of the 10th International Computational Accelerator Physics Conference, San Francisco (ICAP 2009), September 2009.

[6] S. Franke, W. Ackermann, B. Steiner, T. Weiland, J. Enders, C. Hessler, Y. Poltoratska: Implementation of Fringe Field Dipole Magnets into the V-Code Beam Dynamics Simulation Tool, Proceedings of the 11th European Particle Accelerator Conference (EPAC 2008), Genoa, Italy, June 2008.

[7] S. Franke, W. Ackermann, T. Weiland, J. Enders, C. Heler, Y. Poltoratska: Handling Overlapping Fields Within the V-Code Beam Dynamics Simulation Tool. Proceedings of the 2009 Particle Accelerator Conference (PAC 2009), Vancouver, Canada, May 2009.

[8] C. Klose: Entwicklung und Implementierung von Algorithmen zur automatisierten Strahleinstellung am S-DALINAC, Diss., Darmstadt, Germany, 2010.

[9] CST AG - Computer Simulation Technology, Bad Nauheimer Str. 19, D-64289 Darmstadt, Germany, http://www.cst.com.

[10] S. Franke, W. Ackermann, T. Weiland, T. Quincey, F. Hug, P. Görgen, C. Klose, M. Platz: Further Development of the V-Code for Recirculating Linear Accelerator Simulations, Proceedings of the XXV International Linear Accelerator Conference (LINAC 2010), Tsukuba, Japan, September 2010.

A BOTTOM-UP APPROACH TO AUTOMATICALLY CONFIGURED TANGO CONTROL SYSTEMS

S. Rubio-Manrique, D. Beltran*, I. Costa, D. Fernandez-Carreiras, J.V. Gigante, J. Klora, O. Matilla, R. Ranz*, J. Ribas*, O. Sanchez, CELLS-ALBA Synchrotron, Cerdanyola del Vallés, Spain

Abstract

Alba maintains a central repository, so called "Cabling and Controls database" (CCDB), which keeps the inventory of equipment, cables, connections and their configuration and technical specifications. The valuable information kept in this MySQL database enables some tools to automatically create and configure Tango devices and other software components of the control systems of Accelerators, beamlines and laboratories. This paper describes the process involved in this automatic setup.

INTRODUCTION

ALBA is the first Synchrotron Light Source built in Spain[1]. Its 3 GeV Storage Ring has been commissioned during 2011, receiving in 2012 the first users for the 7 beam-lines. Most of ALBA control system [2] have been developed on top of Tango Control System [3][4].

An amount of 5531 devices are controlled up-to-now in ALBA accelerators (linac, booster and storage ring) using 150 control Linux PCs. Those devices are accessed from the control system using several hardware interfaces (ethernet, fiber optics, serial lines, GPIB, PLC inputs, …) that had to be configured in the Tango database.

To populate the tango database, information related to hardware configuration and connections has been imported from ALBA's Cabling and Controls Database (CCDB) [5]. The cabling database keeps information of all hardware connections and the network configuration of all ethernet equipments. We used this information already in CCDB to automatize the creation and configuration of several control subsystems, like vacuum, front-ends, motor controllers and EPS [6]; speeding-up the process and minimizing the errors.

THE CABLING AND CONTROLS DATABASE

The Cabling and Controls database was developed in-house by ALBA management and information software section (MIS) to be used as the main knowledge support for design, control, tendering, progress follow up and a search tool providing multiple views of the system.

This MySQL database has been used since 2007 to keep a knowledge base of all the equipments and devices used at ALBA accelerators and beam-lines. The Cabling database keeps all the information about equipment types, cable configurations, connections between devices distribution of hardware in racks, routing of cables between equipment and network configuration. It also provides fast access to all available documentation for each type of equipment. It actually contains in 346 racks with 6326 equipments of different 638 equipment types. These equipments are connected using 17623 cables of 292 different cable types with a total length of 167,43 Km.

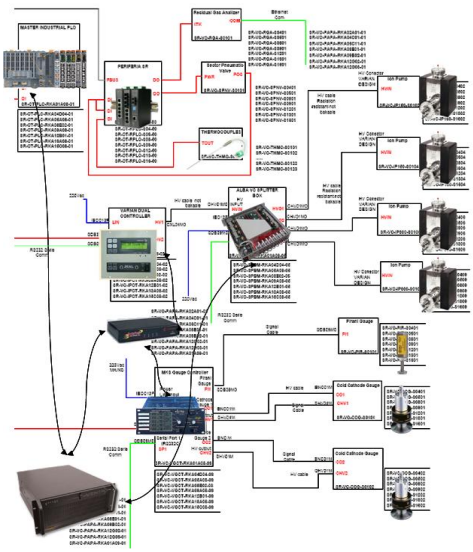


Figure 1: Diagrams of every subsystem have been produced prior to its introduction in our cabling database.

Introducing Cables in the Cabling Database

Filling this database required and intensive effort by a group of engineers (one dedicated to each subsystem). They introduced most of the information in the cabling database during 2008, although installation continues and the database is updated almost daily. All the documentation used to populate the database has been kept attached to each equipment and rack; so it is possible to download original diagrams (Fig. 1) and equipment manuals from the CCDB web client (Fig. 2).

The maintenance of the database is managed by a group of electronic engineers. Modification or insertion of new cables in the database is done using spreadsheet files, that contain source and destination equipment-terminal pairs and cable types. Every file is cross-checked prior to insertion to avoid errors and collision with previous data.

The Cabling Database API

The common method for visualization and modification of information in the cabling database is the CCDB web client. This tool browses all the information in the database and provides many methods for viewing and exporting the data to text files. The first integration with

*On leave

the control system was based on reading these exported files, but soon a better interaction was needed to guarantee a system up-to-date.

Now the CCDB database becomes available to the control system using the CCDB python API. This python module provides methods to search for equipments and get lists of connections, names and network information. Some translation features have been added to the API to store links between every Control and Hardware object, this links will allow every single script to update or cross-check the properties of a Control object from the CCDB.

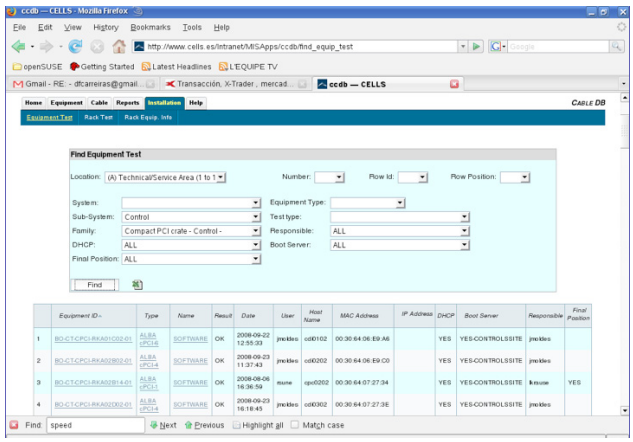


Figure 2: The cabling database web client.

GENERATING A CONTROL SYSTEM

Creating Vacuum Controls Devices

The accelerators vacuum control, with 1496 devices, is the largest control system in our Tango database. Creation of vacuum devices have been done automatically for all subsystems, using the cabling database python API to extract all needed information and all connections between equipments.

For each vacuum device (gauge, pump or valve) that belongs to our accelerators we obtain:

- Port and controller reading the device.
- Serial line and Controls computer connected to the controller
- PLC managing the pressure interlock
- Valves controlled by this interlock.

A Tango database entry is generated for each device and every configuration value. We initially used the same text format as Jive (the default Tango configuration tool), but finally we moved to comma-separated values files. A column based format allowed us more flexibility to add some Tango information like device host and run-level. These methods for massive insertion and modification of Tango database have been added to functional-Tango [7] python module, available from Tango repositories.

The architecture of hardware and software objects may differ, not having a one-to-one conversion between CCDB and Tango. The translation features of the API allow to link a different Tango device with each connector of a patch panel (e.g. serial lines) or link a single complex devices with all the software controllers of its parts (e.g. a single mirror translates into a group of gauges, pumps, motors and thermocouples). This translation allows to rebuild the whole control architecture from the cabling.

Distributing Tango Devices in Control Hosts

The Tango devices (control software processes) declared in our database are distributed in 150 PCs in our racks area. Although the control of most equipments is distributed by location (e.g. front-end devices are controlled from dedicated front-end racks) some subsystems have no dedicated racks and use spare ports of other devices, optimizing the number of devices and length of cabling connections.

Furthermore, every Tango process is assigned to a particular host from the CCDB, following the connection between the gauge in the tunnel to a controller in the rack and then the serial line from the controller to the patch panel of that particular industrial PC where the serial line controller will be configured.

Configuring Devices

There are many parameters that are configured in the Tango control database using the information updated from the CCDB. The most common is the number of ports to be used in those devices that have multiple configurations (serial lines, ADC´s, ion pump splitters), the device configuration will get this value from the CCDB to configure which channels will be used and which will be ignored when connecting to the hardware. Also dynamic attributes will be created for used channels only.

Another example are the sizes of ion pump, this value will be used by the control system when converting the current readings to an estimated pressure value. In some cases the sizes of ion pumps change. For example, when vacuum sections are replaced by insertion devices. These changes must be updated in the Tango database and the information is queried from CCDB.

Information relative to networking (IP, masks, host-names, routing) helped to control or update existing hardware that is interfaced with the control system using TCP/IP. This allows to check and restore the status of these devices (splitters, Icepaps, DAQs,) after a powercut.

Keeping the Databases up to date.

During the last year several modifications have been done in our accelerators. Diagnostic elements have been moved and several vacuum chambers have been replaced by new insertion devices. These changes required to modify racks and cables and introduce the new equipment and connections in the graphical applications.

In case that any change of the routing is done, the technicians will update the information in the CCDB and the control system can detect the change and advice to update it. New CSV files, containing device names and all its property values, are extracted from either CCDB or

Tango databases and are compared to detect missing information, errors or not applied changes.

INTERACTION WITH GRAPHICAL INTERFACES

The information available in the cabling database is used to extend the displays of the control system and allocate equipments in generic browsers. The cabling API enables applications to sort and search by location of controllers or by location of the gauges connected to them (racks and equipments in tunnel don't follow the same name). It allows to group equipments affected by same logics and also make them findable in the racks area when searching for problems.

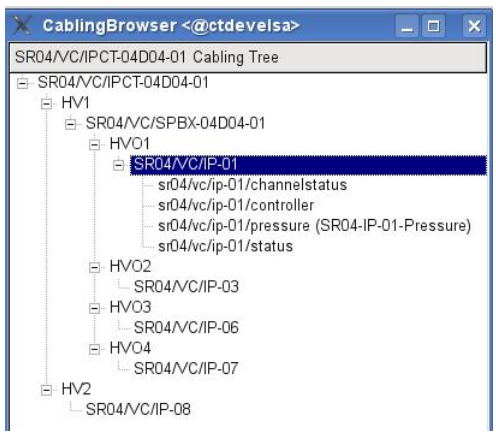


Figure 3: Tango devices can be shown in trees using the physical connections between equipments.

This extra information is added to the GUI's in two ways:

- In the Tango Database, adding cabling names as aliases of devices or labels of attributes (Fig. 4).
- In cabling-enabled widgets, browsing widgets (Fig. 3 and Fig. 5) that translate Tango names into CCDB to group the devices or attributes using their location or connections.

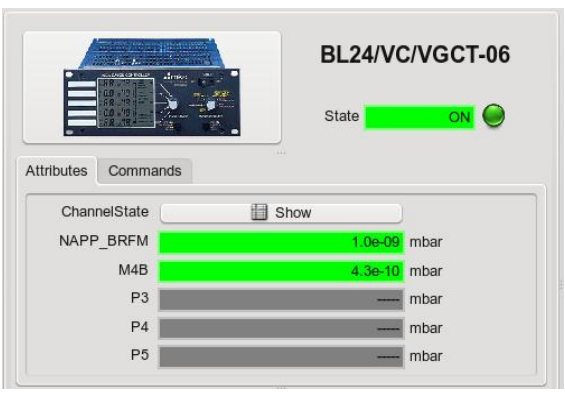


Figure 4: Names of vacuum gauge ports are labelled with the tag of the device connected to them.

When ports of a same controller are used to read different equipments (e.g. an in-vacuum mono-chromator and a mirror separated by valves) the grid widget (Fig. 5) scans the controllers in the vacuum rack and then it obtains the experimental instrument connected to each port. This information is used to reorganize values in panels and assign labels to pressure values.

Figure 5: Labels from cabling are used to group in grids the experimental equipment pressures.

CONCLUSIONS

Automated creation and configuration of big Control systems is a recurrent topic due to its applications in big facilities, reducing the amount of work and human errors when configuring thousands of devices. Acquiring and maintaining a cabling database is hard to do in a running facility, but it is a unique opportunity for new machines as this kind of database is a valuable tool at all levels.

An important application of automated configuration is the fine-tuning of the whole control system, modifying configuration parameters in many devices at the same time (new time-outs in all serial lines, different polling period in different groups of devices, update all host names at once). We used it during commissioning to optimize our control system performance, having also a fast way to do configuration roll-back.

We have now an automated way to load new values to all properties: reducing the time needed to deploy generic systems in beam-lines or adding subsystems once the machine was already operating (insertion devices). A lot of work has been saved and a lot of extra information became available to users.

REFERENCES

[1] D. Einfeld, “Progress of ALBA”, EPAC-2008, Genoa, Italy
[2] D. Fernández et al. “Alba, a Tango based Control System in Python”, ICALEPCS'09, Kobe, Japan.
[3] A. Götz, E.Taurel, J.L. Pons, P. Verdier, J.M. Chaize, J. Meyer, F. Poncet, G. Heunen, E. Götz, A. Buteau, N. Leclercq, M. Ounsy, “TANGO a CORBA based Control System”, ICALEPCS'03, Gyeongju, Korea
[4] http://www.tango-controls.org
[5] D. Beltran, et al. “ ALBA CONTROL & CABLING DATABASE ”, ICALEPCS'09, Kobe, Japan.
[6] D. Fernández et al. “Personnel protection, equipment protection and fast interlock systems.”, Icalepcs'11. Grenoble, France
[7] S. Rubio et al., “Dynamic Attributes and other functional flexibilities of PyTango”, ICALEPCS 2009, Kobe, Japan

AN OPERATIONAL EVENT ANNOUNCER FOR THE LHC CONTROL CENTRE USING SPEECH SYNTHESIS

S. Page, R. Alemany Fernandez, CERN, Geneva, Switzerland

Abstract

The LHC Island of the CERN Control Centre is a busy working environment with many status displays and running software applications. An audible event announcer was developed in order to provide a simple and efficient method to notify the operations team of events occurring within the many subsystems of the accelerator. The LHC Announcer uses speech synthesis to report messages based upon data received from multiple sources. General accelerator information such as injections, beam energies and beam dumps are derived from data received from the LHC Timing System. Additionally, a software interface is provided that allows other surveillance processes to send messages to the Announcer using the standard control system middleware. Events are divided into categories which the user can enable or disable depending upon their interest. Use of the LHC Announcer is not limited to the Control Centre and is intended to be available to a wide audience, both inside and outside CERN. To accommodate this, it was designed to require no special software beyond a standard web browser. This paper describes the design of the LHC Announcer and how it is integrated into the LHC operational environment.

SYSTEM ARCHITECTURE

Announcements are generated on a dedicated server computer running Scientific Linux. The server generates announcement text based upon received data and translates the announcement into an audible message which can then be delivered to clients by a web server. In order to prevent disruption of the operational version of the Announcer, there are separate web servers for the CERN Control Centre and users not on the operational computer network. Figure 1 illustrates the processes involved to create an announcement and deliver it to a user's web browser.

ANNOUNCEMENT SOURCES

There are two sources of announcements at present: the LHC Timing System and software surveillance processes within the control system.

Announcements from the LHC Timing System

The LHC Timing System [1] is a good central source of information about both the current state of the LHC and new events that occur within the machine. As such, it was the first source of events used for the LHC Announcer. A timing receiver (CTRI) is installed in the LHC Announcer server, allowing the reception of timing

Table 1: Announcements from the LHC Timing System

Category	Events
Beam	○ Energy in 0.5 TeV increments ○ Fill numbers ○ Next injection ring, bucket and no. bunches ○ Injection ○ Beam dumps
Collimation	○ Collimators starting and stopping
Experiments	○ Beta* in 1m increments during squeeze
Feedback	○ Start dynamic changes
Instrumentation	○ Wire scans
Mode	○ Accelerator mode changes ○ Beam mode changes ○ Operational mode changes
Post-mortem	○ Global post-mortems
Power	○ Power converters start and abort ramps ○ 60A PC Permit changes ○ Sector Access changes
RF	○ Start and stop dampers

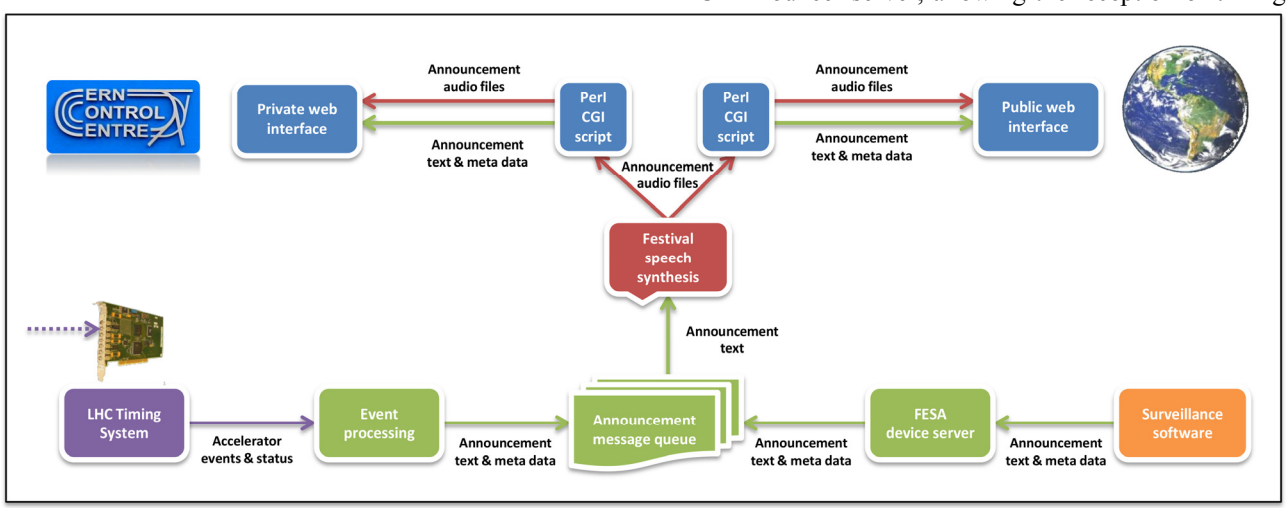


Figure 1: The LHC Announcer architecture.

```
~/public_html>ls events
0_1317273215042118_power_Start power group, 123_.ogg
1_1317273504022752_mode_Beam mode, ramp down_.ogg
2_1317273505598175_beam_Fill number, 2169_.ogg
3_1317273879025657_power_Start power group, 123_.ogg
4_1317276201319233_mode_Beam mode, no beam_.ogg
5_1317279961198602_surveillance_At least one RF cavity, or low level, is down_.ogg
7_1317272524832764_power_60 amp PC permit, cleared for 4,5 and 5,6_.ogg
8_1317272810826116_rf_Stop dampers_.ogg
9_1317272812671035_mode_Beam mode, beam dump_.ogg
```

Figure 2: Example event files with meta data encoded within their names.

events. Each timing event may have a corresponding 16 bit payload, indicating a value associated with the event. Many events within the timing system are not suitable for direct mapping into announcements, so received events are processed by call-back code written in Perl. This allows announcements to be triggered on the arrival of a particular combination of timing events or when a timing event's payload changes by a given amount or passes a configured threshold. For example, the beam energy is received from the timing network at 10Hz. The event processing code will generate an announcement whenever the energy passes through a 0.5TeV boundary or if the energy stabilises at a particular value for more than 10 seconds. A deadband prevents repeated announcements.

Table 1 shows events that are currently announced using data received from the LHC Timing System and the categories to which they belong.

Announcements from Control System Surveillance Processes

After the LHC Announcer had been running for some time using only the LHC Timing System as a source, it became clear that it would be useful to allow other control system processes to make announcements. This was achieved by implementing a device server for the Announcer using the standard controls framework for the LHC (known as FESA) [2]. The LHC Announcer appears as a standard device within the control system, with a single property that triggers an announcement when set. This allows applications to communicate with the Announcer using the standard Controls Middleware (CMW) [3]. In order to limit who may trigger announcements, access to the controls device is protected by the Role-Based Access Control system (RBAC) [4].

Since messages announced by systems surveying the state of the LHC may potentially be sensitive, the FESA interface allows a flag to be set for each announcement to determine whether it should be sent to the publicly visible web server.

Details of the LHC Big Sister software that uses this mechanism to make announcements are given later in the paper.

SPEECH SYNTHESIS

Announcements from the Timing System or FESA server are added to a message queue in text form. A server process reads each message from the queue in turn and passes it to the Festival Speech Synthesis System [5] in order to generate the announcement audio. The audio is stored in an Ogg Vorbis file with the associated meta data (timestamp, category and message text) encoded within the name of the file as shown in Figure 2. Since a burst of events may occur at a faster rate than clients are able to play the audio, the 10 most recent events are stored at any time.

Several synthesised voices were tested when the LHC Announcer was originally developed, with the objective to find one which was both realistic and clear in its enunciation of the event messages. Finally a voice known as 'Nick' developed by the University of Edinburgh was chosen. The use of the synthesised voice required a non-commercial use agreement to be signed.

WEB INTERFACE

In order to make the LHC Announcer easily available to as wide an audience as possible, it was decided to use a web interface. As a result, only a standard web browser is required to run the LHC Announcer [6], rather than a dedicated application.

Features

The LHC Announcer web interface (shown in Figure 3) plays audio announcing any events that occur from the moment that the web page is opened. A rolling history of the last 20 received events is displayed at the bottom of the page, along with the time at which they occurred.

Categories of event (e.g. Beam, Collimation and RF) can be enabled or disabled, allowing the user to select the types of events that they wish to hear. A further option allows events from disabled categories to be included within the event history, in which case they will be shown greyed-out to distinguish them from events that were audibly announced.

A drop-down menu allows the user to add a Vistar fixed display to the web page, allowing the status of the LHC and the other elements of the accelerator chain upon which it depends to be shown visually in parallel to audio event announcements.

Implementation

The web interface uses Javascript to retrieve events and to update the event history and Vistar image (if displayed). When the web page is opened, the current

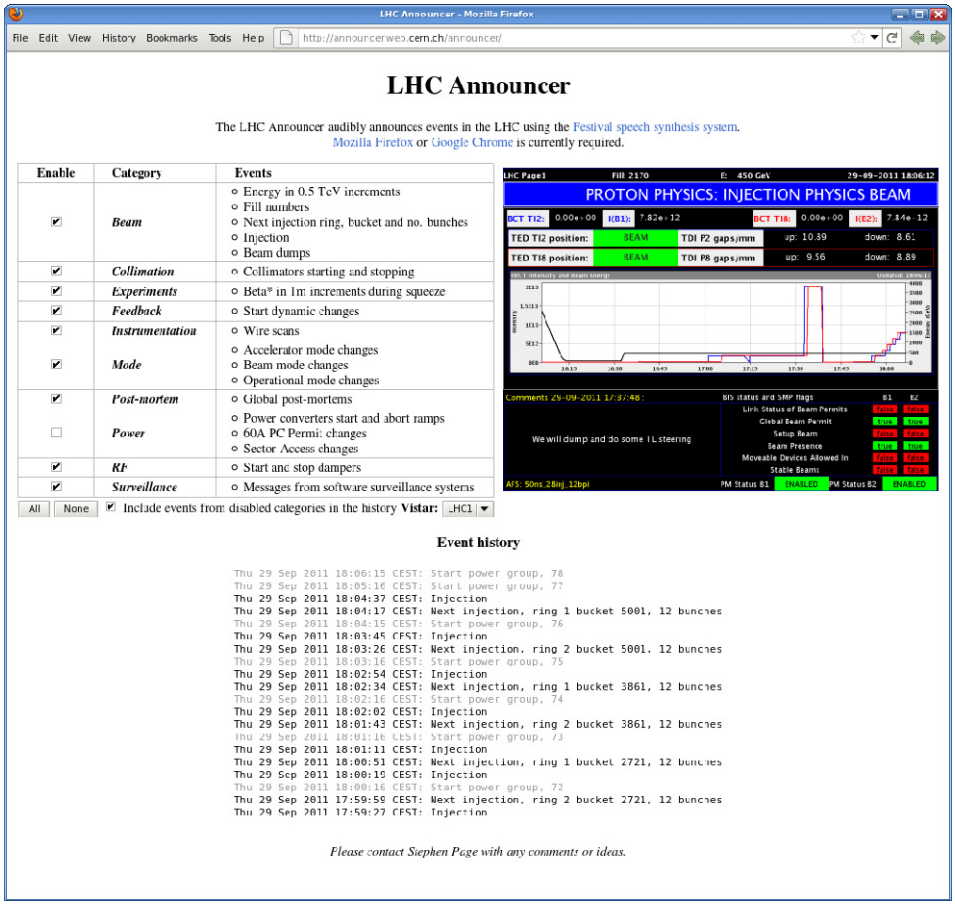


Figure 3: The LHC Announcer web interface.

UTC time is read from the client computer's local clock. That time is then passed in a call to a Perl CGI script which returns meta data for the next of the 10 known events that occurred after that time. The request for new event data is repeated periodically. When a new event is received by the browser, the timestamp of that event is passed with new requests, ensuring that all events are announced, even if a series of events occur faster than the corresponding audio can be played. On reception of meta data for a new event, the web page's Javascript determines whether to retrieve the Ogg Vorbis audio file from the web server and to update the event history based upon the options selected by the user.

This implementation works well when the number of active clients is reasonably low. For systems with a large number of clients, the periodic execution of the Perl CGI script used to retrieve data about new events could become a bottleneck. In this case, it may be more efficient to investigate technologies allowing new announcements to be 'pushed' from the web server, such as WebSockets (though browser support for these is rather immature) or so-called 'long polling'.

Web Browser Compatibility

The LHC Announcer uses relatively recent technologies that require explicit support by the web browser. The web page is written in standard HTML5, which is supported by the majority of modern browsers. However, the HTML5 Specification [7] does not dictate which audio codecs browsers must support for the *<audio>* tag, which the LHC Announcer uses for announcements. Unfortunately, there is no single compressed audio format that is currently supported by all of the most common web browsers, with most of them supporting either MP3 (which has commercial licensing and patent issues) or Ogg Vorbis (an open alternative). Ogg Vorbis is used for the LHC Announcer as it is supported by Mozilla Firefox, which is the most common multi-platform web browser in use at CERN and is also the only web browser available on all of the CERN Control Centre consoles. Google Chrome has also been found to work. Note that if it were important to support all HTML5-compliant browsers, then it would be possible to generate audio in multiple formats, though of course this would involve additional processing overhead.

SURVEILLANCE PROCESSES – THE LHC BIG SISTER

During 2011 a new monitoring infrastructure called Big Sister was put in place for the LHC based upon the architecture of the Software Interlock System [8].

Through a rather complex logic which correlates information from different accelerator systems, the LHC Big Sister can anticipate failures, providing operators with enough warning time and preventing the beams from being dumped, allowing more efficient operation of the LHC.

The logic trees are very flexible and quickly configurable. They can be associated with a measurement of a condition, a device state or property, beam mode, beam quality. The measured parameter is compared to a reference value, with the outcome that the measurement either matches the desired value (condition is true) or not (condition is false). Each of these basic building blocks is associated to one piece of accelerator equipment and they can be combined to produce a more elaborated logical condition. The top-level condition can be either true or false depending on the condition of the elementary building blocks via logic ANDs, ORs or NOTs.

As many alarms and warnings in the LHC Control Room provide visual information, there is always the risk that the operator overlooks some of those alarms or warnings because of a saturation of visual information. Since the LHC Big Sister tries to catch situations that will, in a short while, result in a beam dump, giving the warning messages visually was discarded as an option because failing to react to a warning may result in the loss of the LHC beams. Therefore, the LHC Announcer was chosen as the communication mechanism from Big Sister to the operations team. In this way, the operator does not need to actively seek the information, rather the information comes to him in the form of an audible message. Many beam dumps have been avoided thanks to the combined action of Big Sister and LHC Announcer.

When a condition monitored by Big Sister is false, the associated message is exported to the LHC Announcer which then reports the message and warns the operators about faulty equipment or a forgotten action. The shift crew reacts to the warning, fixing the problem and avoiding a potential dump of the beams. For example, one of the logic trees monitors the status of the Radio Frequency Cavities of LHC. If two or more cavities enter into a faulty state the beams will be dumped. The Big Sister continuously monitors the status of the cavities and as soon one of them enters a faulty state it is announced via the LHC Announcer. Consequently, the operator switches it back on and operation proceeds. Of course, there is also a visual alarm if a cavity becomes faulty, but it has been overlooked on several occasions Since the message has been reported by the Announcer, this type of fault has always been corrected without loss of the beams

LHC ANNOUNCER USAGE

The LHC Announcer has proved useful to a number of different types of user. Its main purpose is to inform operators within the CERN Control Centre of events occurring in the LHC, which may require their attention. Additionally, some equipment experts use the Announcer in order to be informed of events that may relate to tests that they are performing.

The public version of the LHC Announcer web interface has also been popular with people around the world who are interested in the operation of the LHC. As such, it also makes a useful contribution to the publication of CERN's activities and the fostering of public interest.

REFERENCES

[1] Julian Lewis et al, The CERN LHC Central Timing, A Vertical Slice, ICALEPCS 2007, Knoxville 2007.
[2] A. Guerrero et al, CERN Front-End Software Architecture for Accelerator Controls, ICALEPCS 2003, Gyeongju 2003.
[3] K. Kostro, et al, The Controls Middleware (CMW) at CERN – Status and Usage, ICALEPCS'2003, Gyeongiu, Korea, October 2003.
[4] S. Gysin et al, Role-Based Access Control for the Accelerator Control System at CERN, ICALEPCS 2007, Knoxville 2007.
[5] The Festival Speech Synthesis System, http://www.cstr.ed.ac.uk/projects/festival/
[6] LHC Announcer, http://cern.ch/announcer
[7] HTML5 A vocabulary and associated APIs for HTML and XHTML, http://www.w3.org/TR/html5/
[8] J. Wozniak, V. Baggiolini, D. Garcia Quintas, J. Wenninger, Software Interlocks System, ICALEPCS 2007, Knoxville 2007.

PROSHELL – THE MEDAUSTRON ACCELERATOR CONTROL PROCEDURE FRAMEWORK

R. Moser, A. B. Brett, M. Marchhart, C. Torcato de Matos, EBG MedAustron, Wr.Neustadt, Austria
J. Gutleber, CERN, Geneva, Switzerland
J. Dedič, S. Sah, Cosylab, Ljubljana, Slovenia

Abstract

MedAustron is a centre for ion-therapy and research in currently under construction in Austria. It features a synchrotron particle accelerator for proton and carbon-ion beams. This paper presents the architecture and concepts for implementing a procedure framework called ProShell. Procedures to automate high level control and analysis tasks for commissioning and during operation modelled with Petri-Nets and user code is implemented with C#. It must be possible to execute procedures and monitor their execution progress remotely. Procedures include starting up devices and subsystems in a controlled manner, configuring, operating O(1000) devices and tuning their operational settings using iterative optimization algorithms. Device interfaces must be extensible to accommodate yet unanticipated functionalities. The framework implements a template for procedure specific graphical interfaces to access device specific information such as monitoring data. Procedures interact with physical devices through adapter software components that implement one of the following interfaces: (1) state-less or (2) state-driven device interface. Components can extend these device interfaces following an object-oriented single inheritance scheme to provide augmented, device-specific interfaces. As only two basic device interfaces need to be defined at an early project stage, devices can be integrated gradually as commissioning progresses. We present the architecture and design of ProShell and explain the programming model by giving the simple example of the ion source spectrum analysis procedure.

INTRODUCTION

MedAustron [1] [2] is an ion therapy and research centre presently under construction in Wiener Neustadt, Austria. The facility features a synchrotron-based accelerator (Figure 1) with up to 5 ion sources for protons, carbon ions and possibly other light ions. It will provide ion beams with energies up to 800MeV to 5 beam lines, one of which is a rotating proton gantry.

The Procedure Shell Execution Framework (ProShell) is a C# application to automate high level control and analysis tasks for commissioning and during operation. Each task called a procedure implements a standardized procedure interface and is deployed as .NET assembly (shared objects). Key features of the ProShell are:

- Allocating resources on behalf of a procedure.
- Uniform access to system, software and physical devices independent of communication protocols for monitoring and control purposes.
- Reception and visualization of device measurements
- Management of generic procedure lifecycle and custom procedure workflow.
- Parallel execution of multiple procedures
- Automatic procedure execution without user intervention
- Manual procedure execution to step through the procedure specific workflow.
- Provide access to control system services hiding implementation specific interfaces and communication protocols.

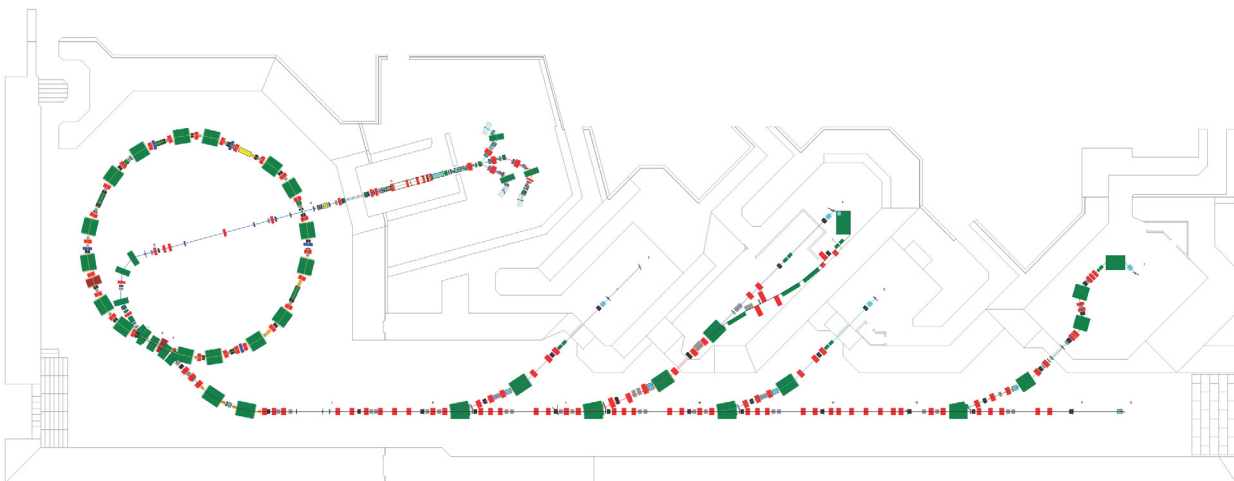


Figure 1: MedAustron accelerator layout.

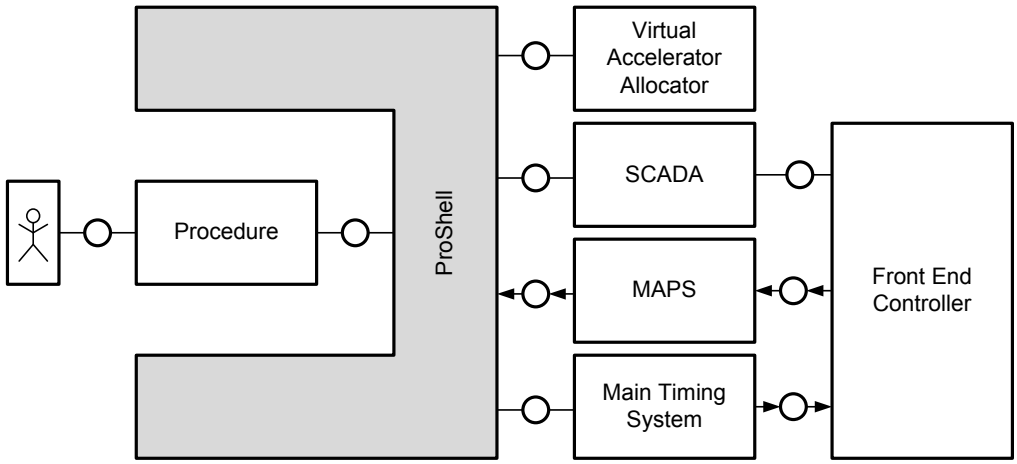


Figure 2: General ProShell architecture and main communication partners.

ARCHITECTURE

Overview

ProShell is a framework to dynamically load and execute procedures implemented as C# classes. As outlined in Figure 2 it provides access to system, software and physical devices for monitoring and control purposes. These services are accessible from ProShell through service-specific *Driver* objects that are used internally and are not directly accessible from the loaded procedures:

- *Virtual Accelerator Allocator* (VAA) is the scheduler of the system that allocates resources for exclusive usage on behalf of a user application.
- *WinCC OA* is a Supervisory Control and Data Acquisition (SCADA) tool from Siemens [5]. It acts as the main communication backbone between user interfaces and procedures on tier 1 and frontend controllers and devices on tier 3 [7].
- *MAPS services* are a set of data servers implementing a publisher subscriber protocol. It forwards measurements and main timing information from front-end controllers to user applications in non-real-time.
- *Main Timing System* (MTS) generates events for beam generation that are delivered to the frontend controllers with a precision of 100ns [4].

In MedAustron the last three services provide a communication channel to the *frontend controllers* running on National instruments PXI crates. Each frontend controller can host multiple *frontend devices* that implement device specific tasks in LabView to control the connected physical device. They also provide a unified interface to WinCC OA through a shared variable OPC server and may emit fast monitoring data through MAPS.

Procedure

A *Procedure* encapsulates specific repetitive control and processing tasks for operation and commissioning written in C#. It is compiled into a .NET assembly (shared library) and implements a common interface to be managed by the ProcedureContext independent of its task.

Procedures may call any available C# functions and libraries. However, procedures should mainly use the API provided by the ProcedureContext to interact with system devices and services.

Procedure Context

ProShell encapsules execution of procedure instances in separate *ProcedureContexts*. Each ProcedureContext acts as a container that provides coordinated access to devices and control system services and manages the procedure lifecycle. Thus it provides the capability to execute procedures in parallel. In case of resource conflicts the VAA will delay the allocation of one procedure and subsequently ProShell will delay the execution of this procedure. As depicted in Figure 3 each ProcedureContext manages the following objects separately:

- *Procedure* implements a specific control or processing task and provides a standardized interface to be controlled by the ProcedureContext.
- *ProcedureLifecycle* handles the general lifecycle of the procedure that is common to all procedures.
- *PetriNetEngine* is a workflow engine that executes the procedure specific workflow defined in a Petri Net Modelling Language (PNML) file [3].
- *DeviceCache* allocates resources on behalf of the procedure using the VAA driver and keeps a cache of C# adapter objects for communicating with the resources.

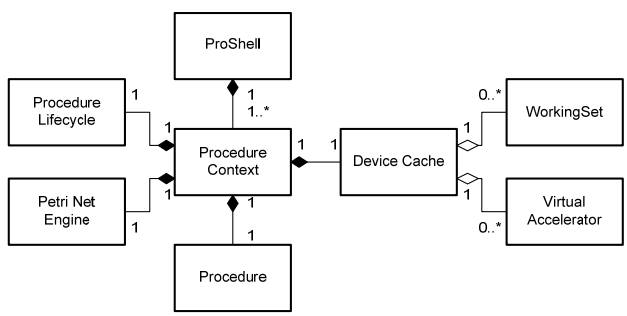


Figure 3: Class diagram for object owned by the ProcedureContext.

Resources

All device instances and types are represented in WinCC OA as data points (DP) and data point types (DPT). In addition each DPT contains a set of data point elements (DPE) that are name-value pairs with a defined value type. Each device implements one of the following interfaces:

- *BasicDevice* is a state-less front-end device interface that only provides a minimal set of DPEs for monitoring.
- *StateDrivenDevice (SDD)* is a state-driven front-end device interface that extends the BasicDevice with additional DPEs to provide an interface for commanding and login to guarantee exclusive access to a device.

WinCC OA also provides two special resource types to control a set of devices concurrently through a single virtual device:

- *Working Set* (WS) is a virtual device that implements the SDD interface and controls a set of SDDs.
- *Virtual Accelerator* (VAcc) controls a set of Working Sets and subsequently a set of devices. It also implements the SDD interface. In addition a VAcc also contains a dynamically assigned Main Timing Generator that allows a procedure to emit timing information for beam generation with an accuracy of 100ns.

Resource Adapters

ProShell encapsulates communication over a number of different communication technologies, shielding procedures from the underlying addressing and communication specifics by providing devices as object that follow the adapter pattern [6].

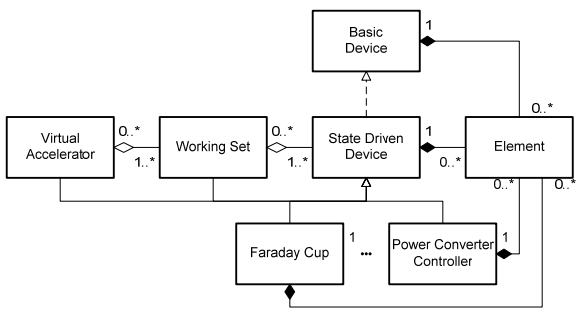


Figure 4: Class diagram of resources adapters for devices, Working Sets and Virtual Accelerators.

Adapter objects provide an object-oriented API that implement BasicDevice or StateDrivenDevice interfaces as shown in Figure 4. Each device exposes a number of elements and functions:

- *Elements* wrap a single name-value pair (i.e. DPE) independent of communication protocol using a specific Driver internally. Classes implementing the element interface provide data type conversion for complex data types and client-side validity checks not supported by WinCC OA.
- *Functions* provide access to multiple Elements that require a specific workflow when reading or writing. In this case the Elements will be available only as protected fields within the device adapter.

General Procedure Lifecycle

The *ProcedureLifecycle* manages the lifecycle part, which is the same for all procedures following the state machine depicted in Figure 5.

When a procedure is opened a new ProcedureContext is created, the procedure specific PNML file is loaded and the state is moved to *Hold*. At this time no resources are allocated.

Subsequently the user can issue the *Initialize* transition to request the allocation of the specified resources from the VAA. The procedure lifecycle state moves to *Ready* when all resources have been allocated.

Thereafter the user may emit an *Enable* transition that parameterizes the procedure specific Petri net and moves the state machine into *Op* state. While in *Op* state the procedure specific Petri net can be executed in steps or run until termination.

The *Disable* transition stops the execution of the PetriNet and the *Finalize* transition releases all allocated resources. If an error is detected, the state moves to *Failed* and a subsequent *Clear* transition stops the petri net and releases all resources.

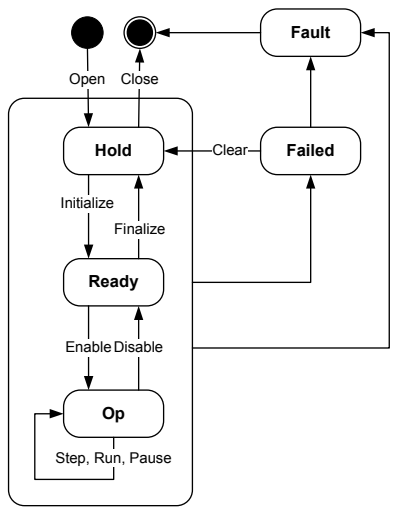


Figure 5: Procedure Context Lifecycle.

Petri Net

The Petri net for each procedure is specified in a Petri net modelling language (PNML) file [3]. The implemented petri net engine extends the standard petri net defined in PNML with the following functionalities:

- Support *coloured tokens* to pass data between transitions.
- Attaching *callback functions* to transitions to execute procedure specific code.
- Extension for asynchronous execution by putting tokens into places programmatically during runtime. This is symbolized in the PNML with asynchronous arcs (dashed lines).

BEAM SPECTRUM ANALYSIS

The ion source beam spectrum analysis procedure detects the particles types generated by a specific ion source. Therefore a current is applied to the bending magnet and the generated field deflects the generated particles depending on the particle mass. The energy of the particles that hit the following faraday cup is measured. Due to the correlation of current to particle type, the generated particles can be detected with peak detection algorithm on the energy over current plot.

```
public override void OnEnable()
{
 _faradayCup.Move(true, 20);
 PetriNet net = Context.PetriNet;
 net.BindParameter("n", (uint)Currents.Cur.Count);
 IPlace place = net.GetPlace("conf");
 foreach (Tuple<double> current in Currents.Cur)
 {
   CurrentToken token = new CurrentToken()
   { Current = current.Item1 };
   place.AddInitialToken(token);
 }
 Context.PetriNet.Trigger();
}
```

Figure 6: *Enable* transition for beam spectrum analysis.

This procedure will allocate an ion source working set containing the power converter for the bending magnet and the following faraday cup during the *Initialize* transition. The subsequent *Enable* transition as depicted in Figure 6 moves the faraday cup into the beam line and configured the *conf* place with n current tokens.

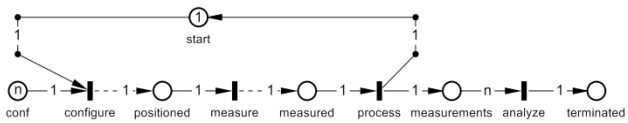


Figure 7: Ion source beam spectrum measurement with synchronous arcs (solid lines) and asynchronous arcs (dashed lines).

After the procedure was brought into *Op* state, the Petri net depicted in Figure 7 becomes active. The Petri net engine will remove a token from *conf* and *start* places and fire the *configure* transition (Figure 8) that applies the current specified in the token from the *conf* place. When the set-point was reached a token will be put into the *positioned* place and subsequently the *measure* transition is fired. During this transition the faraday cup is instructed to perform a measurement that will be put asynchronously as a token into the *measured* place.

```
private void Configure(PetriNet petrinet,
  ITransition transition,
  Dictionary<IArc, Queue<IToken>> input,
  Dictionary<IArc, Queue<IToken>> output)
{
 List<CurrentToken> tokens =
  petrinet.GetTokens<CurrentToken>(input);
 if (tokens.Count == 1)
 {
  var cycleToken = (CurrentToken) token;
  _pcc.CurrentAqn.Subscribe(
   petrinet.GetPlace("positioned"), token[0].Current);
  _pcc.CurrentCcv.Value = token[0].Current;
  petrinet.GenerateTokens(output);
 }
}
```

Figure 8: Petri net callback function for *conf* transition.

After a measurement has been performed for each current token in the *conf* place, n tokens with measurement results will be available in the *measurement* place and the *analyze* transition will be fired. This transition will plot the measurements and performs a peak detection to detect the generated particles.

SUMMARY

This article presented the architecture and selected architecture significant components of the MedAustron Procedure Shell Execution Framework. The application follows a container-based approach where each procedure is executed in its own sandbox. Accelerator devices are controlled and monitored through resource adapter objects that provide an object oriented API to hide communication specific details.

Integration with the main systems (WinCC OA, MTS, VAA and MAPS) has been concluded. First tests have been carried out in the MedAustron test column with the presented procedure and generic procedures that execute beam cycles using physical devices where available and emulating not-available devices through WinCC OA scripts. These tests have shown that generic tasks such as allocation, data conversion and validity checks can be encapsulated in ProShell to provide a simplified interface for procedures and that the Petri net provides valuable feedback for the currently running procedure.

Next steps for ProShell include extending element classes for additional protocols, and specifying and implementing adapter objects and procedures required for commissioning and during operations.

REFERENCES

[1] M. Benedikt, A. Wrulich, “MedAustron—Project overview and status”, Eur. Phys. J. Plus (2011) 126: 69.

[2] M. Benedikt, A. Fabich, “MedAustron—Austrian hadron therapy centre”, Nuclear Science Symposium Conference Record 2008. NSS '08. IEEE, pp.5597-5599, 19-25 Oct. 2008.

[3] J. Billington et al., “The Petri Net Markup Language: Concepts, Technology, and Tools”, Proc. 24th Int. Conf. Application and Theory of Petri Nets (ICATPN’2003), Eindhoven, The Netherlands, June 2003.

[4] J. Dedic et al., “Timing System for MedAustron Based on Off-The-Shelf MRF Transport Layer”, in Proc. IPAC 2011.

[5] P. Golonka, M. Gonzales-Berges, “Integrated Access Control for PVSS-Based SCADA Systems at CERN”, in Proc. ICALEPCS 2009.

[6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, “Design Patterns—Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1995.

[7] J. Gutleber et al., “The MedAustron Accelerator Control System”, Proc. ICALEPCS, 2011.

LASSIE: THE LARGE ANALOGUE SIGNAL AND SCALING INFORMATION ENVIRONMENT FOR FAIR

T. Hoffmann, H. Bräuning, R. Haseitl
GSI Helmholtzzentrum für Schwerionenforschung GmbH, Darmstadt, Germany

Abstract

At FAIR, the Facility for Antiproton and Ion Research, several new accelerators and storage rings such as the SIS 100, HESR, CR, the inter-connecting HEBT beam lines, S-FRS and experiments will be built. All of these installations are equipped with beam diagnostic devices and other components, which deliver time-resolved analogue signals to show status, quality and performance of the accelerators. These signals can originate from particle detectors such as ionization chambers and plastic scintillators, but also from adapted output signals of transformers, collimators, magnet functions, rf cavities and others. To visualize and precisely correlate the time axis of all input signals a dedicated FESA based data acquisition and analysis system named LASSIE, the Large Analogue Signal and Scaling Information Environment, is currently being developed. The main operation mode of LASSIE is currently pulse counting with latching VME scaler boards. Later enhancements for ADC, QDC, or TDC digitization in the future are foreseen. The concept, features and challenges of this large distributed DAQ system will be presented.

INTRODUCTION

At all particle accelerators analogue signals are derived from detectors, rf cavities, function generators, transformers and other sources. A common way to observe these signals in order to understand, trim and operate the accelerator is the readout and signal presentation by oscilloscopes. With an increasing amount of signals at large facilities like GSI or FAIR, this solution gets expensive. Furthermore, it can not be easily integrated into a common accelerator control system, which must provide safe remote operation including data recording, storage and time-correlated presentation.

There are already solutions for analogue signal acquisition and presentation such as OASIS [1] at CERN and ABLASS [2] at GSI. With respect to the high costs of switching matrices and powerful front-end digitizers, the concept of OASIS was rejected as a solution for FAIR. However, long term experience with the low cost ABLASS system has already been obtained at GSI and the reduced performance in resolution compared with oscilloscopes was accepted for most of the signals. The principle of ABLASS is based on the conversion of analogue signals into frequencies, which are then counted in a modular VME system. To provide time-correlated signals acquired over the complete FAIR complex for the future digital control room, the concept of ABLASS will not only be ported to the new control system environment. In contrast to ABLASS the new DAQ system, which is called LASSIE ('Large Analogue Signal and Scaling Information Environment') is designed from the start as a distributed system. As such, LASSIE will be used at all FAIR [3] accelerators, which are part of the modularized FAIR start version: SIS100, HESR, CR, S-FRS, HEBT and pLinac.

GENERAL CONCEPT

LASSIE will be the data acquisition environment for a wide range of beam diagnostic systems. Most diagnostic systems consist of particle detectors like ionization chambers, scintillators, secondary electron monitors or beam loss monitors. Other signals handled by LASSIE include the output of current transformers or direct current measurement from cups and collimators.

However, the concept is not restricted to signals from beam diagnostic detectors. Other arbitrary signals like magnet ramps or accelerator rf-signals can be added for correlation purposes.

The acquisition, analysis and presentation of time correlated signals will provide essential information for beam setup, optimization, control and experiments. This is essential for beam transmission measurement, spill structure analysis and beam loss monitoring.

The system itself will not change or set accelerator hardware i.e. react on beam like feedback systems would do.

For normal operating tasks, it is sufficient to sample the signals with a frequency in the order of 1 kHz. For dedicated beam experiments higher sampling rates (currently up to 1 MHz) will be supported.

LASSIE is based on the Front-End Software Architecture (FESA) [4]. Originally developed at CERN, FESA is now advanced in a collaborative effort between CERN and GSI, allowing for GSI specific modifications. The use of the FESA framework results in a clear separation between the data acquisition part and the graphical user interface part. LASSIE includes both these parts: the data acquisition using FESA and Java based analysis and display tools. In addition, the FESA part will strictly implement the currently emerging guidelines for FESA development at GSI. This will allow the seamless integration of the LASSIE system into the new FAIR control system.

DATA ACQUISITION

LASSIE currently implements one FESA class: the LArge Scaler Array (LASA). This FESA class reads and stores the data of one or more latching scalers as a function of time over one accelerator cycle. With respect

to the high amount of expected channels and the availability of high-performance acquisition modules the VME standard was chosen. All analogue signals not directly suitable for counting are converted into frequencies using I/f or U/f converters. These frequency pulses are counted in VME scaler modules like the Struck SIS3820, a 32 bit and 32-channel multi-scaler. For the time being, a GSI-built timing module decodes the accelerator timing and provides a list of machine events with their corresponding time within the accelerator cycle. This correlates the machine events with the time dependent scaler data.

Modern accelerators have the ability to deliver different beams quasi in parallel. At GSI this is currently done using the concept of virtual accelerators of which there can be up to 16. FESA allows for multiplexed data storage, i.e. data is stored for each virtual accelerator separately. By design, memory allocation within the FESA framework is quite static, although the upcoming FESA 3.0 version allows for a slightly more dynamic memory handling. Nevertheless, as the LASA class is designed to handle 200 or more scaler channels and to store the data over full accelerator cycles of varying length of up to 15 s with selectable sampling frequencies, a more flexible memory handling is required. In addition, the required possibility of selecting sampling frequencies up to 1MHz mandates the ability to disable almost all of the scaler channels but the few ones of interest. This is caused not only by the memory required but also by the bandwidth of the VME backplane, which restricts the data rate. Therefore, the LASA class implements a custom memory management allocating a fixed amount (typ. 50%) of the system memory and is using two memory banks. While new acquired data is stored in one bank, the data of the previous cycle can be read by clients from the other bank. The downside of this implementation is that the LASA class is not truly multiplexed. Data for every accelerator is sent to the clients and it is the client's decision whether to ignore or display it.

Often, beam physics experiments require the storage of data on disk for later offline analysis. This data storage in ASCII or binary form is handled by the FESA class either to a network file-system or to an attached local solid-state disk. The storage procedure is realized in a thread, a so-called concurrency layer, separated from the data acquisition and the full spill data is copied to a memory buffer before writing. This ensures that data acquisition is not hampered by blocking write operations.

In the FAIR control system it is expected that data acquisition classes like the LASA class will also be responsible for setting DAQ related components like amplifiers etc. The LASA class is designed for this task. However, for the time being the class can only obtain these settings from the current GSI control system.

General guidelines for designing the interface of FESA classes at GSI/FAIR are currently being prepared. According to these guidelines FESA classes should return (and accept) physically meaningful values to clients. Using device settings from the current GSI control system, the LASA class can convert the scaler data into the physical input data like currents or voltages. For many diagnostic systems these currents or voltages are proportional to the beam current, i.e. the number of particles per second passing the detector. The proportionality factor however is usually dependent on the energy loss in the detector, which in turn requires information like beam energy, charge state and isotope mass to be calculated. Although these calculations are currently implemented in the LASA class for test purposes, it is still discussed whether this is the appropriate place for such calculations. It requires, that the FESA class is informed of the beam parameters either by the FAIR control system or by the central machine timing system. Both are not available yet.

The LASA class always reads all active scaler channels and stores them in memory. Clients are usually interested only in a few channels. To facilitate easy access to single channels without the need to send all data to the clients, the LASA class uses the ability that a single FESA process running on a front-end computer can present several devices to the clients. Thus, the LASA class running on a front-end computer presents each scaler channel as a single device to the clients. Clients accessing this device can obtain the integrated data over the full accelerator cycle or between two selected events and, if desired, the time dependent spill data also between two selected events. This allows easy use of the LASA class from generic GUIs and other applications of the future control system.

EXPERT TOOLS

In addition to the LASA class, the LASSIE system consists of several expert GUI tools, which are written in Java and are based heavily on CERN libraries for communication with the FESA classes and graphical display.

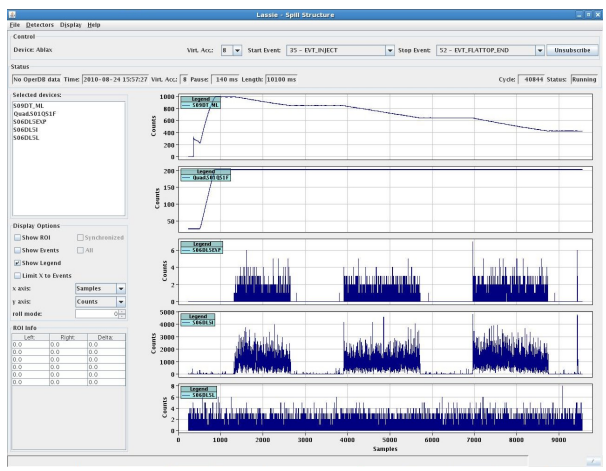


Figure 1: The Lassie spill structure tool for analysis of synchrotron signals (from top to bottom: current transformer, quadrupole ramp and beam loss monitors).

This tool-set contains a general control and monitor tool for the LASA classes running on different front-end

computers, a spill structure analysis tool [Fig. 1], a counter display tool [Fig. 2], a dose level tool [Fig. 3] and others. Using the object oriented approach of Java, the major part of the core functionality, like obtaining data from the FESA classes via suitable providers, storing and analyzing the obtained data and many GUI widgets, could be put into packages common to all tools. This makes the GUI part of the LASSIE system more like a framework and gives it a great flexibility with respect to developing new and dedicated tools. Furthermore, most of the tools are not restricted to use the LASA class. Any FESA class implementing the same interface for the devices it handles could be used. This would for example include the readout of an ADC, which in principle also represents time dependent data.

With the exception of the general control and monitor tool, the expert tools do not address a LASA class (or scaler array) but the devices presented by the running LASA classes. As such, the expert tools inherently can deal with a distributed system, which consists of FESA classes running on different front-end computers in different locations of the accelerator complex.

Of course, the possibility of missing or delayed data from one or more front-end computers must be taken into account. Currently the virtual accelerator number and the UTC timestamp of the GSI machine timing transmitted once per cycle are used to assure the correlation of the data received and to detect any errors. With a first test setup using two LASA systems approximately 300m apart, no significant error rate could be observed for typical SIS cycles of a few seconds.

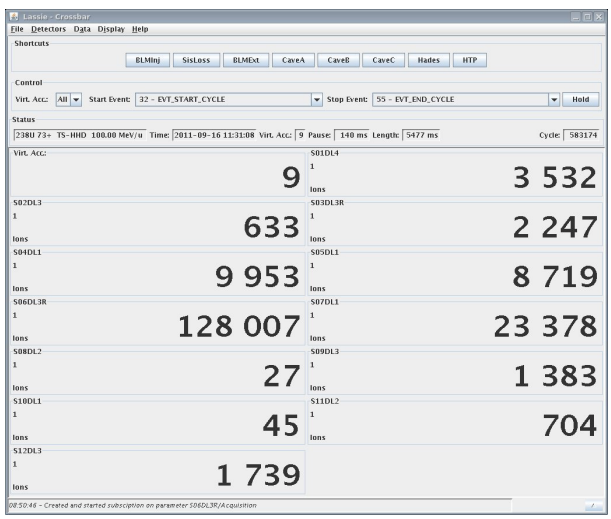


Figure 2: The counter display tool shows the total counter sum of a detector or device after each cycle.

In the FAIR control system the machine timing will be handled via the White Rabbit timing system. This new timing system is based on Synchronous Ethernet, the Precision Time Protocol (PTP), new developed hardware and a field-bus like topology [5]. It provides deterministic data with sub-ns accuracy. Thus, every VME system will be equipped with a White Rabbit timing receiver card.

As every event will be tagged with a time stamp, precise data correlation between the distant crates is assured. In the future this precise tagging may also allow data correlation throughout different timing domains of the accelerator complex.

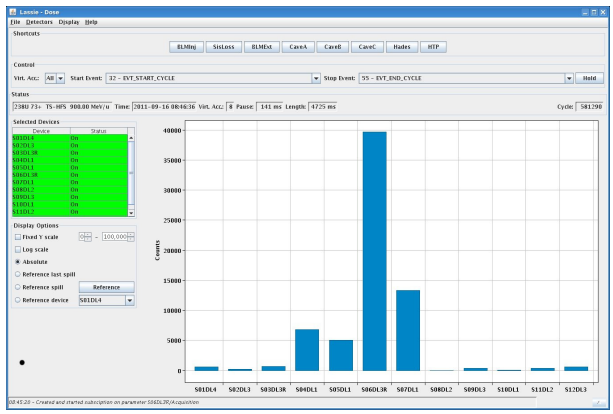


Figure 3: The dose level tool shows the total counter sum of a detector or device after each cycle and is mainly used for beam loss detection.

CONCLUSIONS

LASSIE presents the first implementation of a major beam diagnostic system using the FESA framework and Java based expert GUIs at GSI. It is extensively tested with diagnostic devices from the current SIS18 accelerator of GSI and the current machine timing system. Although many details of the FAIR infrastructure like the new White Rabbit timing system are not available right now, adaptation of the current LASSIE system is expected to be straight forward.

The possibility to extend LASSIE with other FESA classes acquiring time-dependent data e.g. DAC driven magnet power supplies, transforms it into a powerful tool. It will provide oscilloscope-like features fully integrated into the control system, making most of the expensive cables and hardware scopes in the control room obsolete.

REFERENCES

[1] S. Deghaye et al., “Oasis Evolution”, ICALEPCS’07, Knoxville, Tennessee, USA, WPPA04, p. 322, http://www.JACoW.org.
[2] T. Hoffmann, D. A. Liakin, P. Forck, P. Moritz, Proc. 11th Beam Instrumentation Workshop, BIW, Knoxville, Tennessee, USA, p. 294 (2004).
[3] FAIR - Baseline Technical Report, GSI (2006).
[4] M. Arruat et al., "Front-End Software Architecture", ICALEPCS 2007, Knoxville, Tennessee, USA, WOPA04, p. 310-312.
[5] J. Serrano et al.,“THE WHITE RABBIT PROJECT”, TUC4, ICALEPS2009, Kobe, Japan, Oct 2009, p. 93

FIRST EXPERIENCE WITH THE MATLAB MIDDLE LAYER AT ANKA

S. Marsching*, aquenos GmbH, Baden-Baden, Germany
E. Huttel, M. Klein, A.-S. Müller, N.J. Smale, KIT, Karlsruhe, Germany

Abstract

The MATLAB Middle Layer has been adapted for use at ANKA. It was finally commissioned in March 2011. It is used for accelerator physics studies and regular tasks like beam-based alignment and response matrix analysis using LOCO. Furthermore, we intend to study the MATLAB Middle Layer as default orbit correction tool for user operation. We report on the experience made during the commissioning process and present the latest results obtained while using the MATLAB Middle Layer for machine studies.

INTRODUCTION

Synchrotron light sources are an excellent tool for a wide range of studies in biology, chemistry, physics and medicine, as they provide very bright light from hard x-rays to the far infrared.

Most of these light sources are based on electron storage rings, in which electrons circle around at relativistic speed, thus emitting synchrotron light.

For ensuring a maximum performance at the generation of synchrotron light, it is crucial to have tight control over the various operational parameters of the underlying particle accelerator and adapt to changing demands or environmental conditions, like the air temperature.

The MATLAB Middle Layer (MML) [1] is a collection of scripts for the MATLAB programming environment, designed to control and measure parameters of an accelerator. This software creates an abstraction layer enabling the user to write scripts independent of the accelerator's control system. For most uses, even the magnet lattice of the accelerator itself is abstracted. Thus, scripts written for one accelerator can be used for a different accelerator, even if this accelerator has a mostly different design.

Recently, we have adapted the MATLAB Middle Layer for use at ANKA. ANKA is the synchrotron radiation facility of the Karlsruhe Institute of Technology, located in Southern Germany.

In order to use the MML at ANKA, we had to make an interface to the ANKA control system. First we implemented our own control system interface in the MML. Later we migrated to the built-in LabCA support and used a gateway to make data from our control system available via the Channel Access (CA) protocol.

We use some of the existing tools that are part of the MML, like code for orbit response matrix and dispersion measurements as well as the orbit correction code. However, we extended the code for beam-based alignment to provider better fit results and thus improved data for the beam position monitor (BPM) offsets. Finally, we wrote some code to wrap the mostly used functions of the MML in a nice graphical user interface (GUI).

* sebastian <dot> marsching <at> aquenos.com

CHOOSING THE MATLAB MIDDLE LAYER

ANKA has had a software that links the control system to a physical model of the accelerator for quite a long time [2]. However, this software was completely implemented in Java, and thus did not provide the simple scripting mechanisms of the MATLAB Middle Layer. Therefore it was only used for some specific applications (e.g. the orbit correction code still in daily use), but was not used by the accelerator physicists to write their own code.

Learning from this experience, a software for linking the ANKA control system to the MATLAB-based Accelerator Toolbox was developed [3]. Using this software, physicists could finally write simple MATLAB scripts for controlling the accelerator. However, the software suffered from the lack of existing application code. As it was specifically written for ANKA, code from other accelerators could not be used without changes. Thus this software's benefits for daily operation were limited.

Therefore, we decided to adapt the existing MATLAB Middle Layer code for ANKA, reusing parts of the control system link written to connect MATLAB to the ANKA control system. This way, we could link the advantages of having an interface for simple MATLAB scripting with the advantages of already having a number of useful applications.

ADAPTING FOR ANKA

In order to adapt the MATLAB Middle Layer for use at ANKA, we had to create the data-structures specific to the accelerator's lattice and we had to implement a link to our control system.

Creating the Data-Structures

For adapting the MML for a new accelerator some data-structures have to be generated. While their structure is the same for all accelerators and can just be copied, their content (e.g. magnet calibration curves and names of control system properties) has to be customized. The total effort for finishing this task did not exceed a few days.

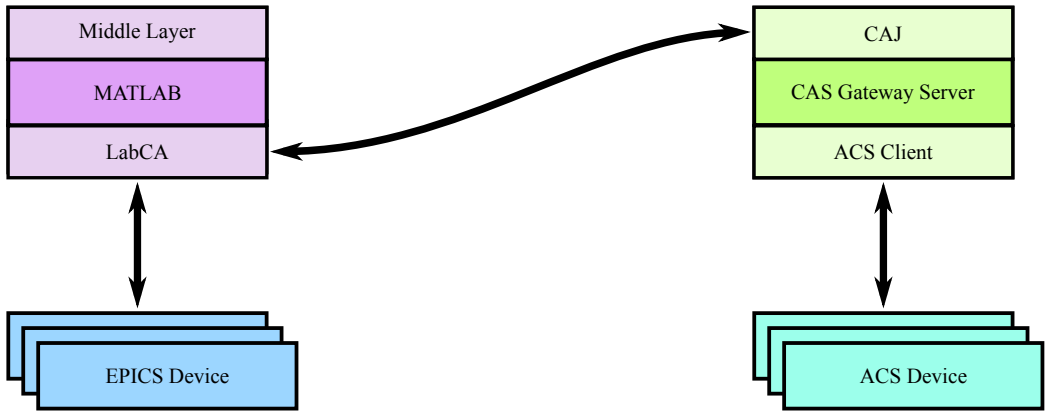


Figure 1: Structure of the control system access for the MATLAB Middle Layer at ANKA. MATLAB communicates directly with EPICS devices using the LabCA library. ACS devices are accessed through the CAS Gateway Server specifically developed for ANKA.

Link to the Control System

At ANKA, most of the devices relevant for the MML (mainly power supplies and beam position monitors) are controlled via the ALMA Common Software (ACS) [4] control system. The MML had no built-in support for this control system, thus we had to develop our own control system link.

As integrating an ACS client directly into MATLAB was not feasible [3], we wrote a server based on the Hyper-Text Transfer Protocol (HTTP) [5]. Each property of the control system would be mapped to a unique resource identifier (URI) identifying this property. A property could then be read by sending a HTTP GET request to the server and written to by sending a HTTP POST request. The actual data would be encoded in the request's or response's body using the JSON protocol [6], a simple, text-based platform-independent protocol for structured data. This solution proved to work well and minimized the implementation effort in MATLAB.

Later, a new beam position monitor (BPM) system based on the Libera Brilliance device manufactured by Instrumentation Technologies was acquired by ANKA. However, this system was not integrated into the ACS-based control system, but uses existing EPICS software developed at DIAMOND [7].

Adding EPICS support was not trivial, because the control system link support of the MML basically depends on two specially named functions (getpvonline and setpvonline). As there is no way for specifying different versions of these functions for different control system properties, we could not use EPICS- and ACS-based devices from the MML in parallel.

We identified two feasible alternatives for integrating both systems in parallel: We could either extend our HTTP server to connect to EPICS-based devices or we could use one of the existing Channel Access (CA) implementations included in the MML and talk to the new BPM devices directly. In the latter case, we would need to use the CA protocol for communication with the legacy control system as well.

We chose the second option for mainly three reasons: First, we would not have another layer of software for the communication with the new BPMs, thus increasing the reliability and performance. Second, using the CA protocol instead of HTTP promised a slightly better performance, because this was a binary protocol specifically designed for control system communication. Finally, we anticipated that in the future more hardware at ANKA might use EPICS instead of ACS and therefore the first argument might become more and more important.

Therefore, we changed our gateway software to provide a Channel Access server instead of an HTTP server, resulting in the setup depicted in Fig. 1. We could reuse major parts of the code communicating with ACS and used the Channel Access for Java (CAJ) [9] library for implementing the server side.

IMPROVEMENTS FOR BEAM-BASED ALIGNMENT

The MATLAB Middle Layer brings a set of functions for running a beam-based alignment (BBA). This is a method to determine the offset of BPMs by changing the magnetic field of a close-by quadrupole for different orbits. These different orbits are set by changing a single corrector magnet. When the orbit does not change when the quadrupole field is changed, the beam is in the center of the quadrupole and thus regarded to be in the center of the close-by BPM.

The quadplot function included in the MML determines the BPM offset by fitting a linear or quadratic function to the orbit change at each BPM for the different position at the aligned BPM. Figure 2 shows the fit for real data measured for BPM S2.01 (horizontal) at ANKA.

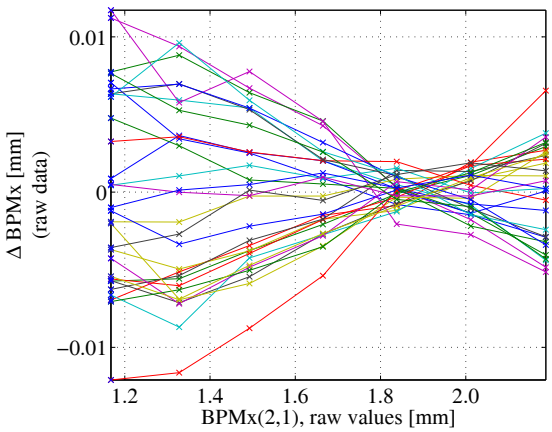


Figure 2: BBA data evaluated with the quadplot function. Apparently the BPM center is at about 1.9 mm. However, due to the noisy data, quadplot miscalculates the BPM center to be at 0.83 mm.

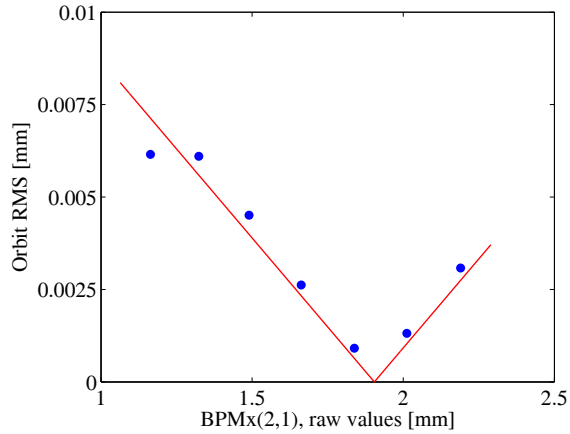


Figure 3: BBA data evaluated with the quadplot2 function. The measured data is shown in blue, the fit is shown in red. Using the same data as in Fig. 2, this function calculates the BPM center to be at 1.90 mm, which is in the expected range.

However, this fitting method is sensitive to single defective values and does not work nicely if the measuring range is far off-center. Therefore we wrote a function called quadplot2, implementing a different algorithm:

This function does not regard single BPMs but calculates the root-mean-square (RMS) of each orbit difference. This RMS value is plotted over the beam position measured at the BPM to be aligned. Subsequently we fit a function of the form

$$f(x) = |x - a| \cdot b, \tag{1}$$

where x is the beam position at the BPM to be aligned, $f(x)$ is the RMS of the orbit difference for the respective orbit, a is the BPM offset and b is a free fit parameter.

The result of the fit for the same data as in Fig. 2 is depicted in Fig. 3. For the example data, this algorithms improves the result, because it is less sensitive to the asymmetry of the measured data and less sensitive to noise.

Still, single data-points used for the fit can be invalid, e.g. because the change for the corrector magnet sent to the control system has not been applied correctly. Therefore, we added an interactive mode, where the user can see the measured data and the fit and select or deselect individual data-points to be included in the fit. In this way, the user can remove obviously faulty data-points and gets an idea of how good the fit results are. The improvements achieved by using this interactive mode for a partly defective measurement are shown in Fig. 4.

USER-FRIENDLY GUI

In order to make the frequently used functions of the MML available to those users who do not want to learn how to use the command-line interface, we created a simple GUI which serves as a starter for these functions. Figure 5 shows the start panel, which then opens sub-panels for

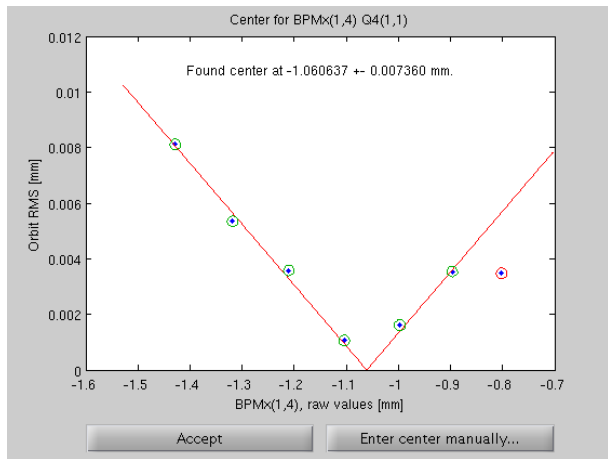


Figure 4: Interactive fitting GUI. The last data-point has been deselected, because it is obviously flawed.

the different functions like dispersion and orbit-response-matrix measurements, beam-based alignment and orbit correction.

This panel is even useful to those users who know the command-line interface, but just want to quickly run a measurement without having to look up the exact syntax of the corresponding function.

THE MATLAB MIDDLE LAYER IN ACTION

As important measurement tools like orbit-response-matrix and dispersion measurements are provided in a standardized way by the MML, it very easy to use LOCO [10], a tool for analyzing the linear optics of an accelerator using orbit-response-matrix measurements. In fact LOCO is already included in the MML distribution and only a few

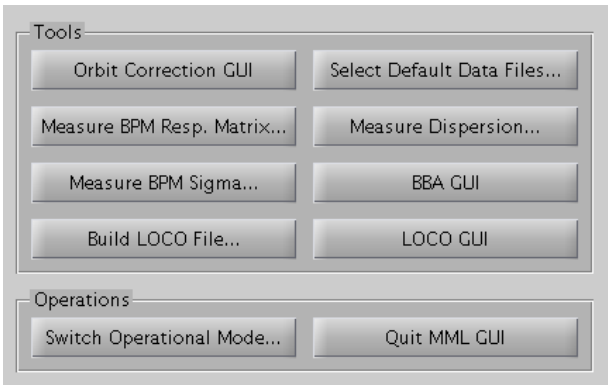


Figure 5: Starter GUI panel for the MML at ANKA.

accelerator-dependent parameters have to be adjusted.

One of our first analyses using LOCO and the MML revealed, that the kick strengths used for measuring the response matrix strongly dependend on the corrector magnet.. The kick strengths fitted by LOCO are shown in Fig. 6.

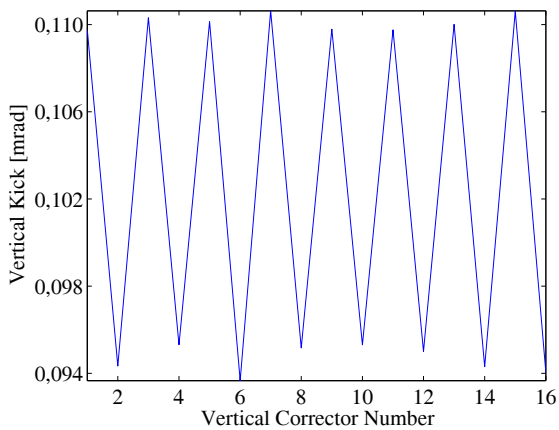


Figure 6: Vertical corrector kicks fitted using LOCO.

First, because of the alternating structure, we suspected numerical noise causing a defective fit. However, closer investigations revealed, that this was caused by an error in the calibration curves converting from magnet current to kick strength and vice versa.

For geometrical reasons, there are two types of vertical corrector magnets at ANKA. Both are expected to have a slightly different kick strength of 1.485 mrad A^{-1} GeV^{-1} and 1.650 mrad A^{-1} GeV^{-1} respectively.

However, as we now discovered using LOCO, in reality both types seem to have the same kick strength of about 1.5 mrad A^{-1} GeV^{-1}.

CONCLUSION

We commissioned the MATLAB Middle Layer for use at ANKA. The commissioning process was quite simple and would have been even simpler if we had used one of the control systems directly supported by the MML.

We now frequently use the MML for machine physics studies and are investigating the option to use an MML-based orbit correction code for regular operation, too. Physicists at ANKA writing their own code for machine physics measurements have already started using the MML to get information about the state of the accelerator.

The alternative BBA code we have written shows significant advantages over the code distributed with the MML in handling imperfect measurement data. The MML simplifies the task of performing a beam-based alignment dramatically compared to our old solution which required the user to copy the measured data into Microsoft Excel for further evaluation.

By creating a GUI for the most commonly used functions, we could even convince those users to use the MML, who are not familiar with the Middle Layer's MATLAB command-line interface.

In summary, adapting the MATLAB Middle Layer for use at ANKA has already brought significant benefits within the first few months of use and we are looking forward to exploit the features of the MML to an even larger extent.

REFERENCES

[1] G. Portmann *et al.*, "An accelerator Control Middle Layer using MATLAB", PAC'05, Knoxville, May 2005, p. 4009, http://www.jacow.org.

[2] I. Kriznar *et al.*, "Cross-Platform and Cross-Accelerator Machine Physics Programs with Databush", ICALEPCS'03, Gyeongju, October 2003, p. 151, http://www.jacow.org.

[3] S. Marsching *et al.*, "A High-Level Interface for the ANKA Control System", ICAP'09, San Francisco, August 2009, p. 318, http://www.jacow.org.

[4] ALMA Common Software website, http://www.eso.org/projects/alma/develop/acs/.

[5] R. Fielding *et al.*, "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC2616, June 1999, http://www.rfc-editor.org.

[6] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)", IETF RFC4627, July 2006, http://www.rfc-editor.org.

[7] M.G. Abbott *et al.*, "The DIAMOND Light Source Control System Interface to the Libera Electron Beam Position Monitors", ICALEPCS'09, October 2009, Kobe, p. 694, http://www.jacow.org.

[8] T. Straumann, "labCA – An EPICS Channel Access Interface for scilab and matlab", May 2008, http://www.slac.stanford.edu/comp/unix/package/epics/extensions/labca/manual/.

[9] Channel Access for Java website, http://epics-jca.sourceforge.net/caj/.

[10] J. Safranek *et al.*, "MATLAB-based LOCO", EPAC'02, Paris, June 2002, p. 1184, http://www.jacow.org.

DEVELOPMENT OF A SURVEILLANCE SYSTEM WITH MOTION DETECTION AND SELF-LOCATION CAPABILITY

M. Tanigaki*, T. Sano, Y. Hirai, M. Miyabe, S. Fukutani, H. Kawabe, Y. Kobayashi, Y. Kuriyama, N. Sato, K. Takamiya, and Y. Morimoto
Research Reactor Institute, Kyoto University, Kumatori, Osaka 590-0494, Japan

Abstract

A surveillance system with the motion detection and the location measurement capability has been in development for the additional surveillance of the facilities in our institute. Each remote box, which consists of a surveillance camera, a GPS unit and a Wi-Fi unit, sends the motion picture along with its location data over the network. The server analyzes such information to detect the unwanted access and sends the status or alerts to the clients. The system is about to be installed for the actual surveillance.

INTRODUCTION

Surveillance cameras and sensors are widely used for the security control in various types of institutes and facilities. Our institute also has surveillance cameras and sensors, which have the primary responsibility for detecting unwanted accesses to our institute. One of the problems in the present surveillance cameras is the lack of the motion detection capability, therefore security agents are required to watch the display throughout the day to find such unwanted accesses. Another problem is the fixed positioning of units because the devices used for the surveillance system are based on hardwiring connections for analog signal of video transmissions. This restriction prevents us from adding temporary cameras for increased securities or for subsidiary purposes. Even worse, the old-fashioned hardwiring system requires locating the position of surveillance cameras by hand. To eliminate such problems, we are constructing a surveillance camera system with motion detection and self-locating features based on a server-client scheme. A monitoring unit consisting of a network camera, Wi-Fi and GPS modules, acquires its location by use of GPS or the radio wave from surrounding Wi-Fi access points (AP), then sends its location to a remote server along with the motion picture over the network. The server analyzes such information to detect any unwanted accesses and serves the status or alerts on a web-based interactive map for the easy access to such information. We report the current status of the development and expected applications of such self-locating system beyond this surveillance system.

* tanigaki@rri.kyoto-u.ac.jp

SYSTEM OUTLINE

The present system is based on a conventional serve-client scheme. A typical configuration of the present surveillance system is shown in Fig. 1. The surveillance is performed by a remote box, which consists of a network camera, a GPS module, a single board computer, and a Wi-Fi access point. All the devices inside the remote box are basically operated by either 5 V DC or 12 V DC, expecting the battery-driven operation. The network connection between the remote box and the server is managed by the local area network in our institute and Wi-Fi connection.

A conventional network camera is in a waterproof housing and placed outside the remote box. This camera can be connected by Power-over-Ethernet technology (PoE) to realize the data transmission and power supply by a single wire. A USB-type GPS module is also implemented in this waterproof housing to ensure the detection of the exact place of the network camera. The USB connection required for this GPS module is prepared by a pair of USB - RJ-45 conversion adapters, which can extend a USB connection up to 50 meters, almost the same distance as that a PoE connection. Therefore, the camera can be placed up to several tens meters away from the remote box, which gives a greater flexibility for the installation.

A linux-based single board PC is embedded in the remote box. The GPS module placed in the camera housing is finally connected to this single board PC for the analysis of GPS location. The NMEA sentences of GPRMC and GPGGA are interpreted to obtain the coordinates of the camera and the time and date of GPS measurement. Obtained location data is sent to the server and used as the position of the camera. This single board PC is also expected to be used as the interface to the devices, such as the control of the floodlight and making sounds for the alert etc.

As for the self-locating feature, the cases that the GPS signals from satellites are not available(e.g. inside or behind the building) should be assumed. To deal with such cases, a self locating technique by using the Wi-Fi signals of surrounding AP is also developed as the alternative way to determine the location. In this technique, the locations of APs are previously determined as the reference points. The number of reference APs varies depending on the places, but at least three reference APs are expected with in the radius of 50 meters. In theory, the intensity of radio wave should follow the inverse squares law, but such simple cor-

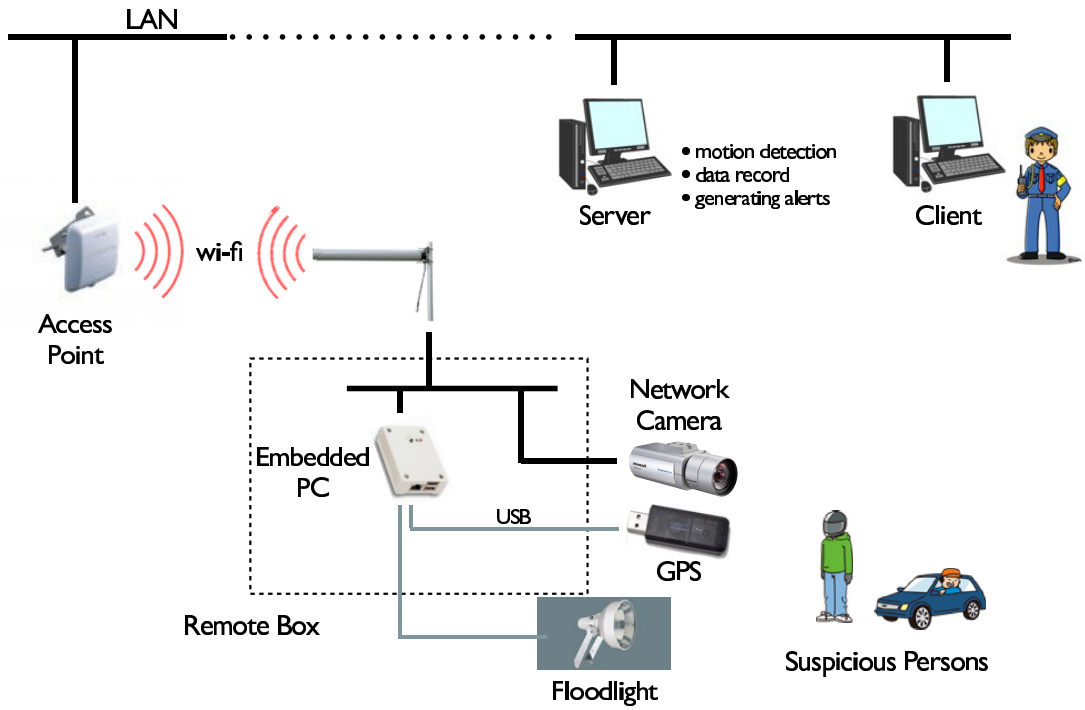


Figure 1: Configuration of the surveillance system.

relation is largely disturbed in real cases by the multipath propagation or reflections caused by the surrounding obstacles. Therefore, the correlation between the signal intensity vs distance are measured and empirically approximated to simple functions (Fig. 2). In the remote box side, the Wi-Fi signals detected are identified by their BSS-IDs, and respective distances among reference APs are determined from their respective signal intensities. The location is obtained as the weighted mean of the coordinates of detected reference APs with distance as weights. In the trials for the cases inside the building, approximate positions of the remote box can be determined in the accuracy of 10 meters.

The visual data obtained by the network camera is continuously sent to the server as streaming data. The server is responsible to detect any unwanted accesses and manages the alerts and responses to such accesses. The motion detection is based on motion[1] and kmotion[2] is used as the default front-end for this system as well as archiving the obtained activities. Additionally, each alert is shown on GIS softwares such as Google Earth in client PCs as an icon at the position corresponding to the latitude and longitude data obtained by the GPS module implemented inside the camera housing. Other responses to an unwanted access can be arranged with the combination of a single board PC in the remote box, such as turning on a floodlight at the camera as a warning to suspicious persons.

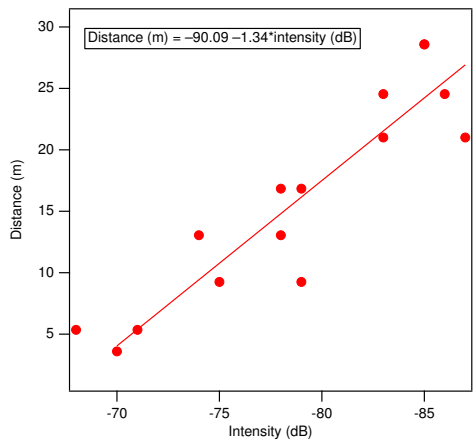


Figure 2: A typical correlation between the distance and wave intensity measured for a reference AP. The correlation is well approximated by a simple linear function. This kind of correlations are measured for every reference AP.

CURRENT STATUS

The developments of each component are almost finished and now the installation of the remote box is now

on the way for the surveillance towards unlawful dumping in the premise border of our institute. In such applications, the surveillance system is often in danger of the destructive behaviors by the suspicious persons who are about to dump the refuses. To avoid such destructive behaviors, the remote box will be placed at a safe place inside the premise of our institute, and only the camera unit will be placed about 30 meters away from the remote box with the help of implemented PoE and USB extension features. Since the camera is set outdoors, the self-locating feature by GPS is used in the present case. Now the adjustment of parameters for the motion detection is on the way because the malfunction caused by the rain drops is not negligible.

KURAMA - A SPIN-OFF OF THIS SYSTEM

The magnitude-9 earthquake in the east Japan and the following massive tsunami resulted in the severe damage at Fukushima daiichi nuclear power plant, i.e., the worst nuclear disaster Japan has never experienced before. One of the contribution of our institute to the recovery from the disaster is to develop KURAMA (Kyoto University RAdiation MApping system) for the carborne γ-ray survey in Fukushima and surrounding regions. KURAMA is installed in an automobile and records the readout of γ-ray survey meter along with the location data obtained by GPS while driving around the region. The readouts are simultaneously transmitted to the observation base in a remote place over the mobile phone network and displayed on Google Earth in realtime (Fig. 3). KURAMA is used as the standard system for the carborne γ-ray survey by Fukushima prefectural government and the Ministry of Education, Culture, Sports, Science and Technology in Japan. Results are available to the public from their websites[3] [4].

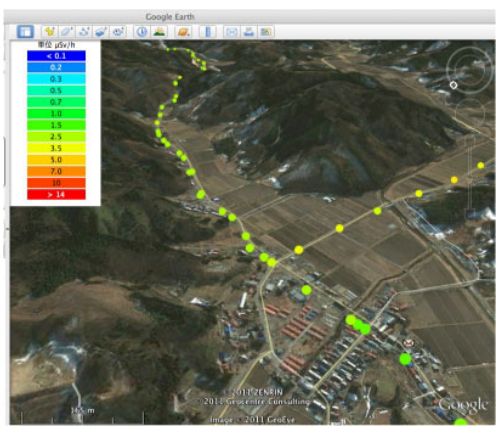


Figure 3: A typical radiation data obtained by KURAMA. The readout of radiation are displayed on Google Earth along with the geographical features.

In the development of KURAMA, some of the techniques which stem of the present surveillance system are introduced. For example, the GPS handling technique used in KURAMA is based on the one in the present surveillance system. Also some techniques developed for KURAMA are imported to the present surveillance system. In KURAMA, the radiation intensity and location of each measurement point are displayed as icons with the respective color on Google Earth. This feature is realized by the dynamic generation of a KML file with the combination of Apache and PHP, which will be used as one of the schemes for the alert in the surveillance system.

REFERENCES

[1] K. J. Lavrsen, *motion*; `http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome`.

[2] Robert E, *kmotion*; `http://www.kmotion.eu/`

[3] *Results of carborne γ-ray surveys in Fukushima prefecture(in Japanese)* `http://www.pref.fukushima.jp/j/soukoukekka.htm`.

[4] *Radiation map by the Ministry of Education, Culture, Sports, Science and Technology in Japan(in Japanese)* `http://denshikokudo.jmc.or.jp/test/mext110628/`.

THE ELECTRONIC LOGBOOK FOR LNL ACCELERATORS

S. Canella, O. Carletto, INFN – Laboratori Nazionali di Legnaro, Legnaro (Padova), Italy.

Abstract

In spring 2009 all run-time data concerning the particle accelerators at LNL (Laboratori Nazionali di Legnaro) were still registered mainly on paper. The electrostatic accelerator TANDEM-XTU and its Negative Source data were logged on a large format paper logbook. For the ALPI booster and the PIAVE injector with its Positive ECR Source a number of independent paper notebooks were used, together with plain data files containing raw instant snapshots of the RF super-conductive accelerators. At that time a decision was taken to build a new tool for a general electronic registration of accelerators run-time data. The result of this effort, the LNL electronic logbook, is here presented.

INTRODUCTION

The LNL Electronic Logbook is a new many-layers software tool; it was put into operation at the end of 2010 after 12 months of work including several plan-do-check-fix cycles.

Here this tool is described in its main components and features. The most significant steps of the long commissioning phase are also described.

CONFIGURATION

An electronic logbook is made of data sets and software programs. It is used to register states, events, texts, images, notes and files according to the shift structures in the working time of the systems to which it applies, here the LNL accelerators and their experimental set-ups. The same tool is used both to register data (to write new data sets) and to retrieve them (to read old data sets) .

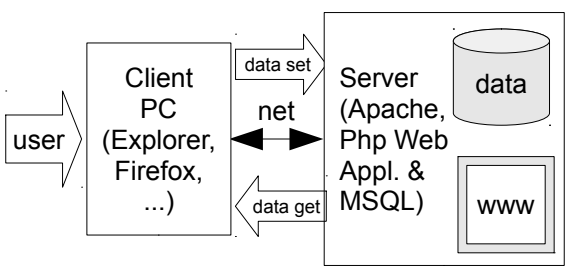


Figure 1: LNL Logbook structure.

The LNL Electronic Logbook has a client-server structure (Fig. 1). A web server application supplies a web interfaces to data. It may be accessed by standard browser (Explorer, Firefox) on any platform (MS Windows, Linux). For a medium and long time range endurance, data are stored in RDBMS (Relational Data Base Management System) structures, At LNL the RDBMS is MySQL in an Apache http server environment. Php procedures dynamically build (on request) the web forms to write or read data from tables in the data base.

Because LNL accelerators are subjected to periodic reconfigurations and rejuvenations, a high degree of flexibility is necessary in the data structures and related web-forms. This was achieved adding to the electronic logbook some intrinsic editing capabilities on the data set configuration and form pages. So the system itself is highly configurable.

General Layout

The main components of the LNL Electronic Logbook are installed on a Linux PC (Fedora) and are:

- An instance of MySQL RDBMS for data table structures and registered data [1].
- The Apache [2] web-server and the related PHP language interpreter, to access data, tables in write and read mode, for configuration and registration.
- AJAX (Asyncronous Javascript And XML) [3] to support dynamic, interactive data throughput from and to the user on the web browser.

Figure 2: LNL Logbook – The main Accelerators Menu.

Data Structure

In LNL Electronic Logbook the data system has a hierarchic, tree structure.

At the top of all there are different electronic logbooks, one for each accelerator (Fig. 2): the electrostatic TANDEM-XTU (with its negative ion source), the ALPI booster and the PIAVE injector (with its positive ECR ion source). In each specific logbook, the root of data tree relies on the calendar experimental shifts list.

The calendar of experiments contains the list of experimental shifts, on a start-end dates basis, plus some attributes for each shift, where these last two terms mean also periods reserved for tests, accelerators maintenance, set-up or shut-down. In the table of shifts, which is updated every 3-6 months, the configured fields are: a code (unique), a spoke-person, start date, end date, experimental set-up, beam-line, beam type (ion specie), ion mass (ion isotope), required energy, energy range (if different energies are required), beam current, accelerator configuration, notes (Fig 3).

All accelerators registrations are therefore associated to the specific accelerator and to its shift defined by the registration date.

On the time side, data registrations are of two kinds: day registrations and hour registrations. The first are written, generally, in the first hours of the day by the operators' team working between midnight and 7 A.M. This set of data may be deleted, updated or changed all the day long and is frozen at midnight. The second set of data (hour registrations) are automatically linked to the current day registration, and to the due shift, too. Hour registration data may be deleted, updated or changed only by the operators who wrote them and only up to the following hour registration, which, automatically, freezes the preceding one.

A special user (administrator) has the privileges necessary to delete or correct any data set at any time.

Besides conventional data sets, specific for each logbook, i.e. integers, decimals, time and string values, attached files, in LNL Electronic Logbook two other tables are foreseen to help shift accelerator logging: Annotations and Announcements. Annotations are free text registrations linked to the current day. They must be one a day for each operator, may be inserted, changed or deleted all the day long but only by its author. Its purpose is to better explain, in textual form, special events, to make comments to current data, or textual amendment to frozen data.

Annotations are always tight to current day data and current accelerator configuration. They may also be characterized by an alert flag that makes them always visible for a given (user settable) amount of time.

This mechanism may be particularly useful for important and/or urgent exchange of messages among operators.

Announcements are similar, but are general purpose and not specifically tight to a given accelerator (logbook) or day registration: they are more free and intended for general messages exchange on accelerators status and necessities. There is no limit in writing these entries.

System Access and Use Modes

The LNL Electronic Logbook is installed on a Linux PC in the LNL data network and there is a limited set of users that may access and use it: the TANDEM-PIAVE-ALPI operators' team, the source team, the experiments coordinator, the administrator.

There are three levels of privileges:

1. administrator
2. calendar coordinator
3. operator

The administrator may do everything, i.e. read, write, create and delete any data, text and defined structure. He is responsible of logbook configuration (defining fields and related attributes included in each logbook) and of DB data maintenance for serious data mistakes that may not be fixed simply by annotations.

Table 1: DB Objects Sizes.

Type of object	n. of items
Accelerators (logbooks)	3
Configuration Tables	7
Data Tables	14
Total n. of Tables	21
Total n. Records in 12 months	~1500

The calendar coordinator has the responsibility of filling the experimental shifts calendar, so determining the associations between accelerators registrations and physic experiments. He only may write, modify or delete current (in terms of date) and future experimental shifts (but not the old ones, which are automatically frozen for him also, after midnight).

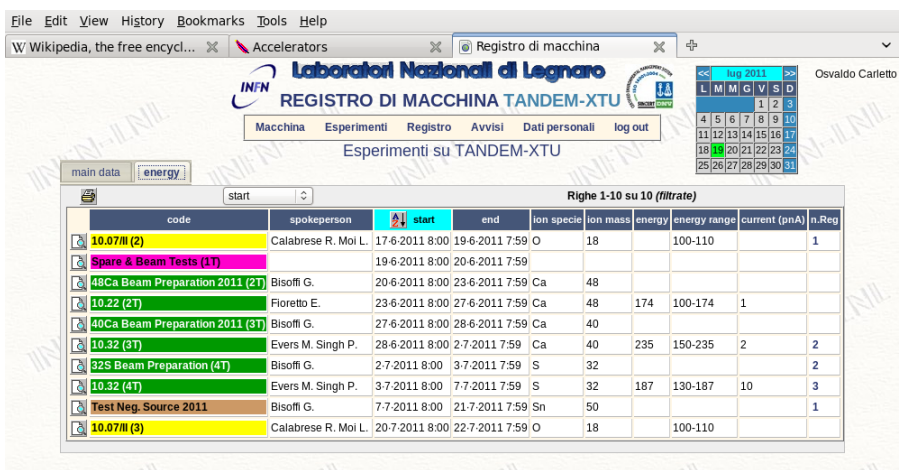


Figure 3: The list of experimental shifts for the TANDEM accelerator.

Figure 4: The external tank of the LNL 15 MV Tandem electrostatic accelerator with photos of an old paper logbook and of the new system.

The operators may read everything, i.e. the calendar tables and registered data of all days and hours in all logbooks, but they may write, delete and change only current day registration and only last hour data (and only if they are the authors of that registration).

Any defined logbook user, no matter which privileges he has, may access in read/write mode to his personal data (name and password).

FINAL NOTES

At the end of 2010 three debugged logbooks have been configured: TANDEM-XTU, ALPI and PIAVE.

TANDEM-XTU (Fig. 4 shows the external tank of the accelerator, its operators' team an photos of its old and new logbooks) is a one-to-one mapping of the old paper logbook into MySQL tables and records.

Differently from the previous, ALPI and PIAVE logbooks both include only a set of essential data and a number of attached files to describe the minimal run-time configuration description of the two accelerators.

The commissioning of the system took place in two steps. A first amount of debugging, especially on table and user privileges and data read/write permissions was performed in June 2010, with the help of two summer students. A second and finer set of check-and-fix cycles took place in autumn 2010 when the system was delivered to LNL operators. At this time a large amount of data configuration or re-configuration, in addition to the final debugging, through proper and intensive use, was performed, together with some deep re-design of the experiments calendar structure.

At the end of 2010 the system was largely debugged and stable.

ACKNOWLEDGEMENTS

The first acknowledgement is for ing. R. Tecchio, who patiently implemented and fixed the DB configurable structure and php-ajax procedures of LNL Electronic Logbook. Then the work of A. Gnec and A. Turcato was of great help for the first debugging phase. Dott. P. Posocco gave us useful suggestions in the fist planning step of the work, while C. Ur is the final proposer of the actual structure and format of the experiments calendar table.

At last all the LNL accelerators operators have to be thanked for their collaboration in the final debugging phase of the system.

REFERENCES

[1] P. Dubois “MySQL, 4th Edition” Addison-Wesley Professional, 2009, http://www.mysql.com/

[2] "About the Apache HTTP Server Project" Apache Software Foundation, http://httpd.apache.org/

[3] J. J. Garrett "Ajax: A New Approach to Web Applications". AdaptivePath.com, http://www.xul.fr/en-xml-ajax.html

OPERATIONAL STATUS DISPLAY AND AUTOMATION TOOLS FOR FERMI@Elettra*

Claudio Scafuri, Sincrotrone Trieste, Trieste, Italy

Abstract

Detecting and locating faults and malfunctions of an accelerator is a difficult and time consuming task. The situation is even more difficult during the commissioning phase of a new accelerator, when the plants are not yet well known. Faults involving single devices are easy to detect, however a fault free machine does not imply that it is ready to run: the definition of malfunction depends on what is the expected behavior of the plant. In the case of FERMI@Elettra, in which the electron beam goes to different branches of the machine depending on the programmed activity, the configuration of the plant determines the rules for detecting malfunctions. In order to help the detection of faults and malfunctions and to display the status of the plant, a tool , known as the "Matrix", has been developed. It is composed by a graphical front-end which displays a synthetic view of the plant status grouped by subsystem and location along the accelerator, and by a back-end calculation engine. The graphical front-end gives also the possibility, once a problem is detected, to focus on its details. The calculation engine is composed by a set of objects known as Sequencers. The calculation rules have been determined by analyzing the various subsystems and global working of the accelerator with plant and operations experts. The Sequencer is designed so that it can also issue commands to the plant. This will be used in the next releases of the Matrix for actively switching from one accelerator configuration to another.

INTRODUCTION

FERMI@Elettra is the new 4th generation synchrotron light source currently under commissioning in Trieste, Italy[1]. It is based on a 1.5 GeV electron linac and seeded free electron laser photon generation. The FERMI@Elettra controls system [2] is based on Tango and is presently made up of about 2800 different Tango devices. The control room operator can access the plant by means of about 1600 different instances of control panels and tens of Matlab applications. These numbers show that the plant is large and complex and can be quite difficult to manage also for experienced users. Another source of complexity comes from the fact the the electron beam can be brought through different paths according to the task that must be performed: several spectrometers for measuring the energy and other characteristics of the beam at different acceleration stages, two bunch compressors, and finally two different undulator chains.

In order to help detecting faults and misconfigurations of the plant, we designed and implemented a tool - known as the "Matrix" - to give a comprehensive and simple view of the state of accelerator to the control room operators. The Matrix analyzes many accelerator components - interfaced by means of Tango devices - and checks if they are ready to transport the beam to a certain destination. The analysis is done mostly utilizing the *State* of the tango devices and the desired *Scenario*.

States

Every Tango device exports a standard *State* variable. The *State* variable can assume a set of pre-defined values: ON, OFF, FAULT,STANDBY, etc.... By reading this variable we can know wether a device is working or not or if has any problem. Further diagnostics of possible problems is done by means of specific, device dependent informations. The value of the *State* variable is calculated from the operating conditions of the equipment that the Tango device is controlling. All the Tango devices in FERMI@Elettra use the *State* variable consistently.

Scenario

The *Scenario* is a coherent collection of rules used to check that the accelerator is ready to transport the beam to a desired destination, such as one of the various electron spectrometers, or one of the two undulator chains. Each rule is in charge of a well defined section and subsystem of the accelerator. Many of the rules are re-used in different scenarios, reflecting the fact the the electron beam must go through the same sections of the accelerator. The *Scenario* is selected and activated by the control room operator.

GRAPHICAL INTERFACE

The graphical interface (see Fig. 1) is horizontally divided in two areas. The upper area display the accelerator Matrix, the lower area shows a diagram of the accelerator and in formations about the selected scenario. The Matrix displays the state of the accelerator by means of a number of "coloured lamps". The lamps are organized in tabular format. Columns are associated to accelerator sections and ordered according to the beam path. Rows are associated to different sub-systems of the accelerator equipment: magnets or power supplies, diagnostics, vacuum, RF, laser. A lamp of a cell thus shows the status of a family of devices of a certain section of the accelerator. The colour of the lamp indicates if these devices are working as expected (green) or not (red). The lamp can be also grayed out when the

*This work was supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3

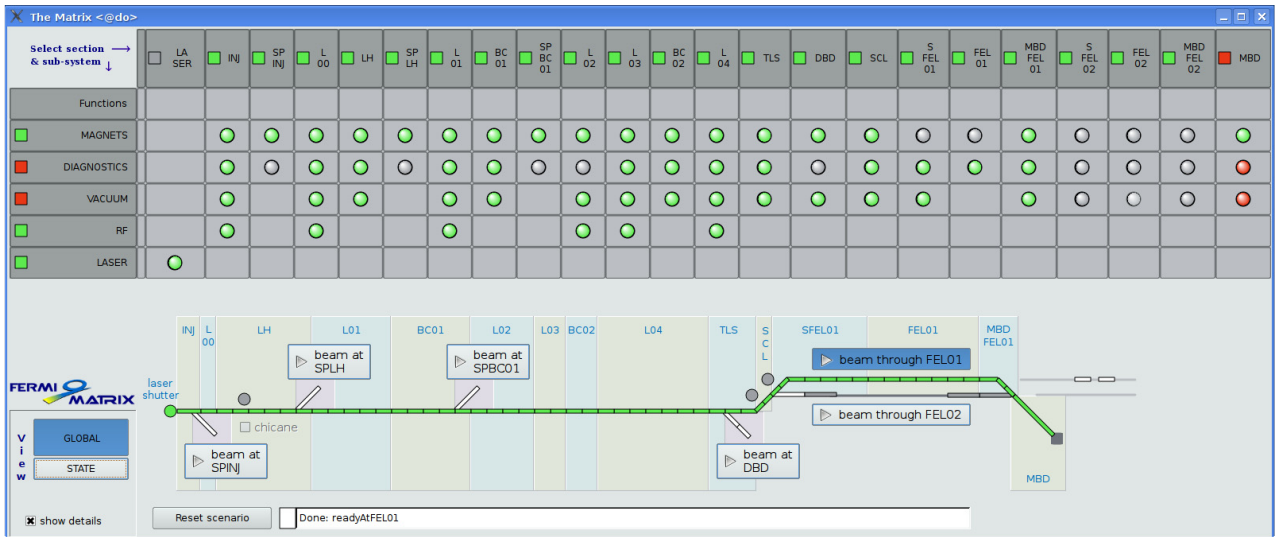


Figure 1: The Matrix panel.

cell is not taken into account in the active *Scenario*. Row and columns headings are also coloured. Their colour summarizes the state of the accelerator section or sub-system according to the active *Scenario*.

Diagnosis of Problems

In case of a problem, signaled by a red lamp, the operator can click on it and check which are the devices that do not behave as expected (see Fig. 2).

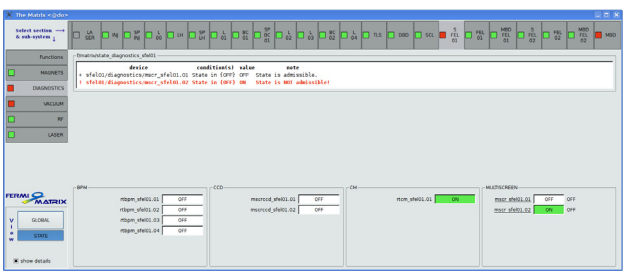


Figure 2: Diagnosis of a problem.

The view is switched to a table that emphasizes the rules which are not met. The state of all the devices involved in the calculations is also displayed in the lower part of the table. By clicking on the state label is then possible to start the control panel dedicated to the specific device. With just two mouse clicks most of the problems or misconfigurations are detected and explained.

Accelerator Diagram

The lower part of the graphic interface displays a schematic diagram of the accelerator and a number of buttons for selecting and activating the different *Scenarios*; the button corresponding to the active *Scenario* is highlighted. The schematic of the accelerator is divided in segments. Each segment is linked to a charge sensitive instrument (BPM, Charge Monitor), so that when the beam is transported through the instrument the corresponding segment is coloured in green.

SEQUENCER

The Sequencer is the building block of the calculation engine used by the Matrix. It is developed in Python starting from the *SequencerBlock* class which derives from standard Python threading.Thread class. The *SequencerBlock* class overrides the run() method (this is required by the Threading class; the run method is called indirectly by the the start() method of the Threading class), implementing the activity diagram shown in UML format in Fig. 3. The run() sequence codifies and formalizes the standard steps that must be done in order to verify equipment conditions or actively modify them; it is foressen in fact that Sequencers will be used also to automate the operations of FERMI@Elettra such as start-up, shutdown, changing of beam path, etc. . .

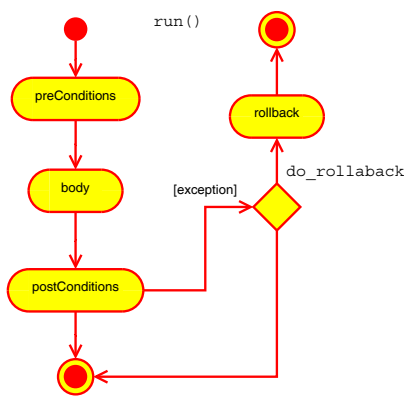


Figure 3: Calculation Engine: main sequence diagram.

The design of the Sequencer comes from many years of practice and experiences of accelerator operations automation [3], which have been thoroughly reviewed and upgraded to object-oriented programming concepts and techniques. The Sequencer has been designed from the start to be easily integrated with the Tango control system.

preConditions, body, postConditions

The Sequencer action is logically divided in three steps, which are called by the run() method. First, check if there are conditions to start the action; this is done in *preConditions* method. Then execute the effective action in the *body* method: calculate the State (that is colour) of a lamp of the Matrix, or issues commands to equipment to change the operating conditions of the plant. Finally check the outcome of the previous steps; this is done in the *postConditions* method.

Errors and Recovery

The three steps:*preConditions*, *body*, *postConditions*, notify errors by rising exceptions which are caught and handled by the *run* method. The Sequencer also provides a *rollback* method which can be used to try to restore the initial conditions in case the *postConditions* are not checked. The call to *rollback* must be explicitly enabled by the programmer of the Sequencer.

Implementation of Real Sequencer Objects in Python

The Sequencer provides a base class and a pattern; the effective knowledge (or intelligence) of the plant that must be checked by a real instance of the Sequencer must be coded by the programmer and is provided by machine and operations experts. The programmer must always derive new Sequencer object from *SequencerBlock* class. The new Sequencer implements directly the three main methods and can add other methods which can be useful for the specific situation. At the same time the programmer can use already existing Sequencer objects by aggregation. Sequencers are written in Python. With Python the programmer can very easily load end execute some Python module at runtime. In this way the programmer can modify the behavior of a Sequencer object at runtime. This feature has been extensively used in the Matrix for changing the behaviour of the program when a new *Scenario* is selected.

MATRIX SEQUENCERS

Several Sequencers have been developed for the Matrix. They perform two main tasks: first, calculate the state of Matrix cells, rows or columns headers: they are derived from *StatesBlock*; second, manage the setting of *Scenarios*: these objects are derived from *ScenarioBlock* (See Fig.4).

StatesBlock

The *StatesBlock* class is a sequencer that analyzes the Tango *State* variable of a group of devices and calculates a resulting *State*. A *StatesBlock* object is configured with a list of Tango devices and their respective admissible states. The *StatesBlock* body method periodically samples the configured devices and analyzes if the group is in an admissible state. The *StatesBlock* contains also some auxiliary methods to report in detail the result of the analysis. The report is used by the Matrix when the detailed view is selected. All the Matrix cell states are calculated by dedicated specializations of *StatesBlock* (e.g. diagnostics_dbd in Fig.4) *StatesBlock* is also the base class for *HeaderStatesBlock* objects which are used to calculate the State (that is the colour of the lamp) for a column or row of the Matrix. *HeaderStatesBlock* use simple syntactical rules to automatically create the list of Tango devices and admissible States.

Deployment

The Matrix needs one *StatesBlock* object for each of its active cells. All these objects are managed by a dedicated Tango device server written in Python, which exports a simple and clean interface for interacting with a *SequencerBlock* object. This deployment scheme makes all the stuff needed by the Matrix generally available as standard control system components. The most important technical reason for this deployment scheme is that it ensures that there is exactly one instance of each of the configured

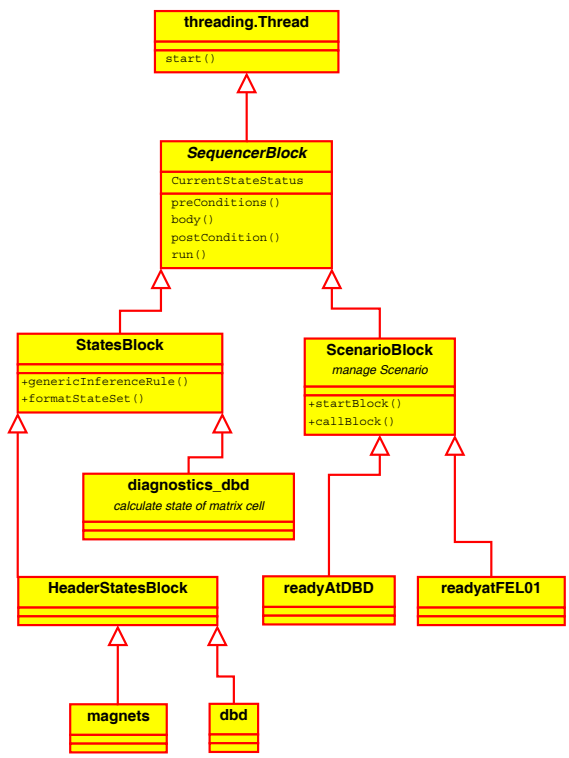


Figure 4: Calculation Engine: class diagram.

SequencerBlock running in the control system of the plant: in this way, when the *SequencerBlock* objects is be used to modify the plant configuration they will do it in an ordered and consistent manner.

ScenarioBlock

Objects of the *ScenarioBlock* class manage the activation of *Scenarios*. A *ScenarioBlock* is programmed with the list of Tango devices and the corresponding *StateBlocks* objects that must be activated for the given *Scenario*. The configured Tango devices are loaded with the chosen *StatesBlock* and started when the *ScenarioBlock* is activated. The *ScenarioBlock* class is derived from *StatesBlock* so that we can reuse the Tango State analysis and reporting features.

CONCLUSION AND OUTLOOK

The Matrix has been used during the commissioning shifts and has shown to be an effective tool for monitoring the general status of the plant. The effectiveness of the Matrix depends strictly on the quality and completeness of the Scenarios: a good collaboration and mutual understanding between machine experts and programmers is of paramount importance in order to correctly and completely analyze and define the various rules. We expect to improve and extend the Scenarios and the diagnostics capabilities of the Matrix with the knowledge gained during the commissioning of FERMI@Elettra.

The next important development of the Matrix will be the automation of operations. As the new accelerator will start operations for FEL light users, standard procedures will be defined for setting up the plant and guaranteeing optimal and repeatable operating conditions. These procedure will be analyzed and implemented by means of *SequencerBlock* objects and supervised by the Matrix. The ultimate goal is to make FERMI@Elettra operable by a team of two control room operators.

THANKS

The initial design and development of the system described in this paper was done by Fabio Asnicar.

REFERENCES

[1] S. Di Mitri, “Commissioning and Initial Operation of FERMI@Elettra”, IPAC 2011, San Sebastin, September 2011.

[2] M. Lonza et al, “Status Report of FERMI@Eelettra Control System”, ICALEPCS 2011, Grenoble, October 2011

[3] D. Bulfone, F. Potepan, C. Scafuri, “Automating ELETTRA Operation with ‘One Button Machine’ ” 17Th Particle Accelerator Conference, Vancouver, Canada, 12 - 16 May 1997, p. 2467

DETECTOR CONTROL SYSTEM FOR THE ATLAS MUON SPECTROMETER AND OPERATIONAL EXPERIENCE AFTER THE FIRST YEAR OF LHC DATA TAKING*

S. Zimmermann, Albert Ludwig University Freiburg, Germany
A. Polini, INFN, Bologna, Italy
G. Iakovidis, E. Ikarios, K. Karakostas, S. Leontsinis, E. Mountricha,
National Technical University, Athens, Greece
R.G.K. Hart, NIKHEF, Amsterdam, NL
S. Bressler, E. Kajomovitz, S. Tarem, Technion, Haifa, Israel
G. Aielli, Università di Roma II Tor Vergata, Rome, Italy
M. Bindi, University and INFN, Bologna, Italy

Abstract

Muon Reconstruction is a key ingredient in any of the experiments at the Large Hadron Collider LHC. The muon spectrometer of ATLAS comprises Monitored Drift Tube Chambers (MDTs) and Cathode Strip Chambers (CSCs) for precision tracking as well as Resistive Plate (RPC) and Thin Gap (TGC) Chambers as muon trigger and for second coordinate measurement. Together with a strong magnetic field provided by a super-conducting toroid magnet and an optical alignment system a determination of muon momentum with high precision up to the highest particle energies accessible by the LHC collisions is provided.

The Detector Control System (DCS) of each muon subdetector technology must efficiently and safely manage several thousand of LV and HV channels, front-end electronics initialization as well as monitoring of beam, background, magnetic field and environmental conditions. This contribution describes the chosen hardware architecture, which as much as possible tries to use common technologies, as well as the implemented controls hierarchy. Emphasis is given to reviwing the experience from the first year of LHC and detector operations, and to lessons learned for future large scale detector control systems.

INTRODUCTION

The Muon Spectrometer forms the outermost layer of the ATLAS [1] detector, as shown in Fig. 1, covering a rapidity range up to $\eta = 2.7$. It comprises four different detector technologies who have been chosen for optimal performance, as either trigger or precision tracking muon chambers, and according to particle rates. Track reconstruction is based in the full muon spectrometer except the innermost region of the endcap ($2.0 < \eta < 2.7$) on in total 380000 **M**onitored **D**rift **T**ubes (MDTs) of 3 cm diameter and up to 6m in length, which are assembled into in total 1150 MDT chambers. Chambers are equipped with an alignment system to in situ monitor and record chamber deformation and relative displacements between chambers, which is required to achieve the design track reconstruction resolution of 50 μm per muon station. In the region of highest background rate, 32 **C**athode **S**trip **C**hambers (CSCs) take the role of tracking detectors. CSCs are multi-wire proportional chambers whose wires are oriented in radial direction and with a segmented cathode. Track reconstruction is done by a fit to the charge distribution, resulting in a track resolution per plane of $< 60\mu$m.

Resisitive **P**late **C**hambers (RPCs) are used as trigger chambers in the muon barrel; ATLAS RPCs are built as a doublet of gas gaps, each formed by 2 bakelite plates separated by a 2mm spacer. Perpendicular readout strips on both sides of the gas gaps allow 2-dimensional track coordinate determination and by requiring a particle track to fall into a predefined geometrical road efficient and fast triggering on muons with a defined momentum. In the endcap region, **T**hin **G**ap **C**hambers (TGCs) are used as trigger chambers; TGCs are multi-wire proportional chambers with a wire-to-wire distance smaller to the wire-to-cathode distance and operated in quasi-saturated mode. In total there are 3588 TGC units, forming 3+2+2 layers in each side's big wheel and 2 layers in the small wheel. Some operating parameters of the 4 detectors are summarized in Table1.

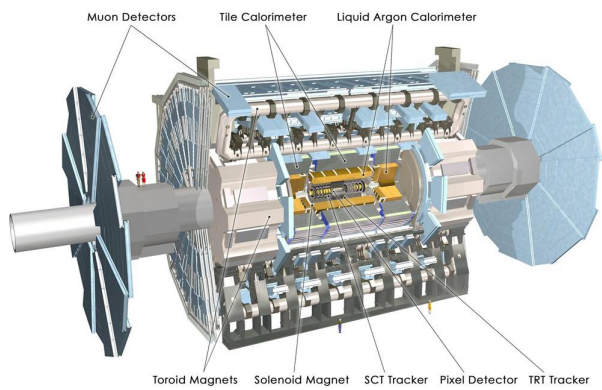


Figure 1: Schematic view of the ATLAS Detector. Elements shown in light blue belong to the Muon Spectrometer which forms the outer most layer of the detector and is split into a cylindrical Barrel part and a set of disk like wheels known as Endcaps.

* On behalf of the ATLAS Muon Collaboration

Table 1: Important Operating Parameters for the 4 Detector Types used in the Muon Spectrometer

	#Chambers	Gas Mixture	Nom. HV	#Readout Ch.
MDT	1150	Ar:$CO_2$93 : 7	3080V	380k
CSC	32	Ar:$CO_2$80 : 20	1800V	31k
RPC	544	$C_2H_2F_4$:i-C_4H_{10}: SF_6 94.7:5.0:0.3	9600V	360k
TGC	3588	CO_2:n-Pentane 55:45	2800V	320k

MUON DCS

The primary tasks of the Muon DCS are

- Controlling the detector power system
- Monitoring environmental conditions and accordingly adjusting operating parameters to maximize efficiency
- Monitoring parameters like voltages, temperatures and RPC trigger rates of the on-chamber electronics
- Monitoring the detector gas system and reacting to changed or abnormal situations by eg adjusting the detector HV
- Configuring the MDT and TGC frontend electronics
- Reacting on information from the DAQ, reinitializing chambers when needed (MDT)
- Controlling/reading out the optical alignment system
- Archiving all relevant information on the detector status as needed for physics data analysis and to trace problems

Power System

For CSC LV, power supplies from Wiener's [2] Maraton series are used; for all other muon LV and HV the commercial EASY[1] system from CAEN [3] was chosen. CAEN's EASY solution is based on a master-slave architecture with a controlling mainframe which houses a set of branch controllers, which each act as master for a chain of up to 6 EASY crates which in turn house the actual LV and HV boards. Crates and boards are compatible with operation in magnetic field and under radiation conditions as those present in the ATLAS experimental cavern during beam operation, while the mainframe and branch controllers are located in a non-hostile area accessible also during beam operation. Table2 summarizes equipment type and the number of devices in use. The RPC system differs from the other muon subdetectors in so far as it contains CAEN ADC and DAC boards in addition to LV and HV types. ADC boards are used for all RPC environmental monitoring and for measuring HV currents on individual gap level; DAC channels are used to control the threshold for the RPC frontend electronics, a function being taken by a different hardware for MDT and TGCs as explained in the next section. The choice of components in the RPC DCS was driven by the strategy to have a single hardware technology, the CAEN EASY solution, to interface to DCS [4].

[1] **E**mbedded **A**ssembled **Sy**stem

Table 2: Muon CAEN Power System Equipment

	Type	# Devices	# Ch.	
CSC	A3540AP	12	144	HV
MDT	A3540AP	204	2448	HV
	A3016	32	192	LV
	A3025	113	452	LV
	A3486	20	40	AC/DC
RPC	A3512AP	49	294	HV
	A3009	80	1280	LV
	A3025	100	400	LV
	A3801	50	6400	ADC
	A3802	24	3072	DAC
	A3086	35	70	AC/DC
TGC	A3535AP	126	4032	HV
	A3025	74	296	LV
	A3050D	24	48	LV
	A3100	74	74	LV
	A3486	30	60	AC/DC

ELMB Based Monitoring And Control

Besides the power system, Muon DCS relies heavily on so called **E**mbedded **L**ocal **M**onitoring **Boards** (ELMBs), which have been custom developped for use in ATLAS and other LHC experiments. Each ELMB, which is magnetic and radiation tolerant, contains a 64 channel 16-bit ADC, 18 general digital IO lines, 8 digital inputs and 8 digital outputs, plus a CAN interface to communicate with. For a detailed description of the ELMB please refer to [5] and [6]. Muon DCS uses around 1300 ELMBs, in a version known as MDM or **M**DT **D**CS **M**odule with a custom firmware and motherboard in the MDT system and around 1500 ELMBs in the TGC system.

MDT MDMs are used to monitor environmental conditions, reading out approximately 14000 temperature probes installed on the MDT chambers, 1650 3D hall probes for determination of the magnetic field map in the ATLAS toroid magnet region and more than 50000 voltage and temperature values from the MDT frontend electronics cards. The MDM in addition handles the initialization of the frontend electronics by downloading configuration parameters to the boards via JTAG. More details can be found in [7].

In the TGC system, ELMBs supply more than 8000 thresholds for the frontend electronics ASD chips, readout and monitor around 3700 chamber displacement sensors and 1600 temperature probes as well as configure and monitor over 23000 on chamber ASICs. A dedicated firmware

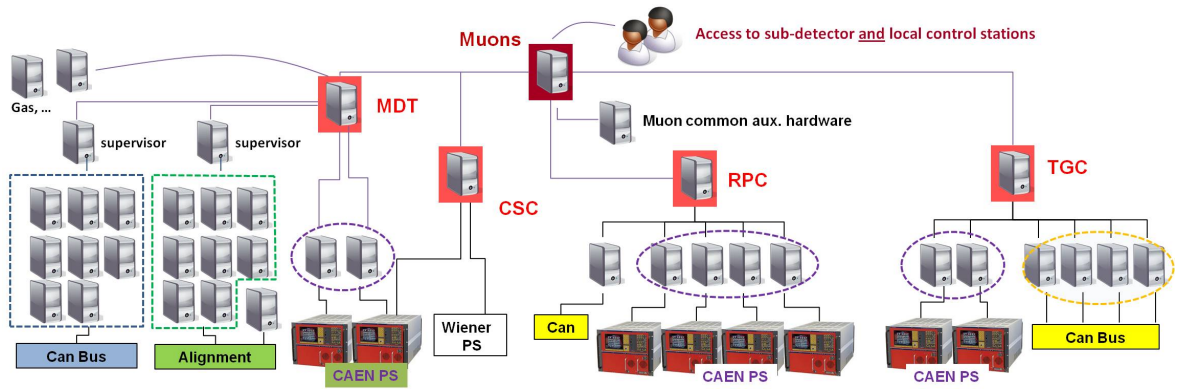


Figure 2: Muon DCS computer hierarchy. MDT MDM and barrel alignment systems have an additional supervisor layer between device layer and subdetector control station. In this the muon DCS system differs from any other ATLAS subdetector control system.

together with custom TGC DCS-PS boards allows detailed monitoring of the chambers' charge spectrum and thus low level information on the detector performance itself. For further details, refer to [8].

MDT Optical Alignment System

The third set of hardware handled by the Muon DCS system is the optical alignment system [9],[10]. Optical alignment is based on monitoring and ananalyzing patterns imaged onto CCD cameras along so called optical lines. Within MDT chambers, alignment lines monitoring deformations consist of a patterned mask, a light source, a lense and the CCD camera. The assembly is known under the name RASNIK [11]. The system is implemented following a hierarchical structure with 3 layers, achieving a high level of multiplexing; acquisition of images is done using 8 PCs and commercial framegrabber video cards. In the endcap the RASNIK technology is complemented by so called BCAM modules, readout in this case is via dedicated multiplexers implemented as VME modules. The results of the image analysis in both cases are stored in a database and used in muon track reconstruction to correct for changes in muon chamber positions.

DCS Software and Architecture

The Muon DCS controls layer has been implemented using the commercial SCADA system PVSS [12] as integrated part of the overall ATLAS DCS system. Components of the common JCOP framework [13] have been used where possible. A total of 40 PCs, running both Windows (XP and 2003) and Linux (SL5) as operating system is used for running the DCS software layer, following a hierarchical architecture (Fig. 2) with a separation between a low-level device layer handling communication with the various hardware and a higher layer of supervisors and so called sub-detector control stations, one per muon technology, dedicated to user interactions and combining information from the different parts of the system. In 2010 an additional layer and node 'MUON' was added which combines information from the various muon sub-detectors and allows unified/common operation; this was driven by ATLAS shift operations moving towards combining shift tasks.
CAEN and Wiener hardware is interfaces to PVSS via the OPC servers provided by the manufacturer; ELMBs are controlled via CanBus from Kvaser PCI cards; for interfacing the Can communication to PVSS the ATLAS developed CanOpen OPC server is used in case of MDTs and a custom PVSS driver in case of TGCs.

OPERATIONAL EXPERIENCE

Power System

The CAEN power system hardware overall is performing as expected, with a board failure rate, usually on individual channel level, well below the 10% level allowed for in maintenance and spares planning. An initial high failure fraction for the TGC A3535 type HV boards was traced to an underdimensioned electronics component which makes the boards very sensitive to any excessive heat e.g. in case of a failure of the rack cooling. Additional checks and actions have thus been added to DCS to detect such situations and react by turning off concerned equipment.
One type of problem still seen from time to time and without yet an optimal way found to handle is boards for which the output voltage V_{out} exceeds the set voltage V_{set}, in some cases by several 100V for HV channels and occassionally exceeding even the hardware voltage limit. 2 cases of broken wires in the CSC system are believed to have been caused by such overvoltage. A second issue are occassional losses of communication with individidual EASY boards, mostly after a power cut, which in the beginning required an access to the ATLAS experimental cavern for a manual reset. To adress this problem, at the beginning of 2011 the so called 'Muon CAEN Reset Network' has been implemented which allows a DCS controlled remote reset of boards. An extension to the reset of branch controllers is planned for the shutdown at the end of this year.

One unexpected finding was that the CAEN OPC Server Event Mode, which became available with CAEN's OPC server version 3, turned out as unsuitable for the use for both MDT and TGC systems[2], due to the large number of HV channels which undergo ramping all at the same time. During the ramp phase the large amount of data updates overloads the OPC server leading to crashes and data loss. Reverting back to the previous polling mode and tuning extensively the OPC group configuration an acceptable and stable situation was found; For the future and possible hardware upgrades, improving the sustainable refresh intervals for HV and LV readings via the OPC chain and PVSS seems however highly desirable. In addition, rare but recurring cases of a full hangup of the communication with the power system have been and are observed. To deal with this a set of watchdog and alert mechanisms has been implemented which proved to allow spotting such problems immediately.

ELMB based Functionality

Experience with the ELMB based part of Muon DCS is very good with an excellent reliablity of the used hardware components. In the full muon system only 2 out of more than 2500 ELMBs failed up to now; both were of MDT MDM type. In the first case the communication with the node via CAN was no longer working; in the second only the ADC part was affected.
The MDT frontend electronics initialization done via JTAG from DCS works very well. With a scheme were string download to the chamber electronics is carried out in parallel on all 96 CanBusses in the MDT system the full detector can be reinitialized in approximately 2.5 minutes. Further improvements are under discussing by storing configurations in the MDM's memory and handling sending the predefined configuration to the MDT chamber electronics by an updated MDM firmware. This would eliminate the current need for sequential string download to chambers on the same CanBus.

Operator Interactions and User Interface Layer

Substantial efforts have been spent during the last year in unifying user interfaces as exposed to non-expert shifters, among other things by developing a set of libraries for a uniform geomtrical representation of the the 4 muon subdetectors and status information on their various components. As an example, the top layer Muon Shifter UI panel is shown in Fig. 3.

CONCLUSIONS

The first full year of LHC operations has proven the chosen design for the Muon DCS a good one, both with respect to hardware and software architecture. No major problems or showstoppers have been found. Feedback gained from shifter operations has lead to several additions and additional efforts of unification, a work which is still ongoing.

[2] no such problems are present in the RPC system

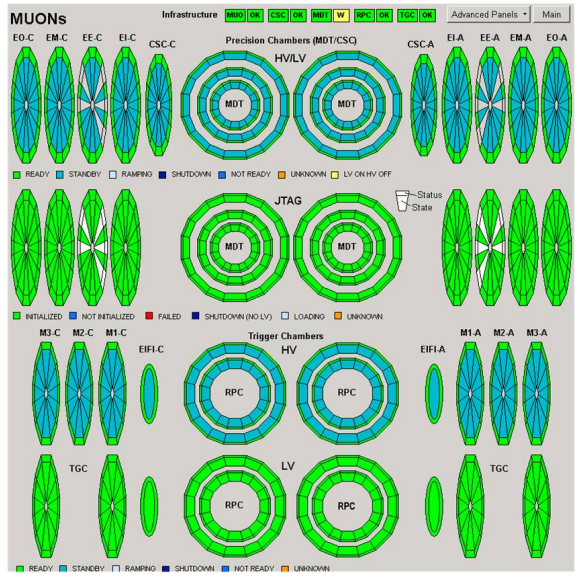


Figure 3: Example of a Muon shifter User Interface panel.

REFERENCES

[1] ATLAS Collaboration, JINST 3 (2008) S08003.

[2] http://www.wiener-d.com.

[3] http://www.caen.it.

[4] G. Aielli et al, "The ATLAS RPC detector control system: Problems, solutions and new oppunrtunities", Nucl.Instrum.Meth. A602 (2009) 796-800.

[5] B. Hallgren et al, "The Embedded Local Monitoring Board (ELMB) in the LHC front-end I/O control system", Proc. 7th Workshop on Electronics for LHC Experiments, Stockholm, Sweden, 2001.

[6] H. Boterenbrood et al, "The Development of the Embedded Local Monitoring Board (ELMB)", Proc. 9th Workshop on Electronics for LHC Experiments, Amsterdam, NL, 2003.

[7] R. Hart et al, "The ATLAS MDT Control System", Proc. ICALEPCS 2009, Kobe, Japan.

[8] S. Tarem et al, "Detector Control System for the ATLAS Muon Endcap Trigger", IEEE Trans. Nucl. Sci. 52 (2005) 1207-11.

[9] J.C. Barriere et al, "The alignment system of the ATLAS barrel muon spectrometer", ATL-MUON-PUB-2008-007, CERN, Geneva, 2008.

[10] S. Aefsky et al., "The Optical Alignment System of the ATLAS Muon Spectrometer Endcaps", JINST 3 (2008) P11005

[11] H.L. Groenstege, "The RASNIK/CCD 3D aklignment system", ATL-MUON-94-063, CERN, Geneva, 1994.

[12] ETM professional control GmbH: PVSS, http://www.etm.at.

[13] O. Holme et al, "The JCOP framework", ICALEPCS2005, Geneva, 2005.

MULTI CHANNEL APPLICATIONS FOR CONTROL SYSTEM STUDIO (CSS)

K. Shroff#, G. Carcassi, BNL, Upton, Long Island, New York, USA
R. Lange, HZB, Berlin, Germany

Abstract

With the development of the ChannelFinder[1][2] directory service and the PVManager[3] client library, a new set of applications have been developed for Control System Studio(CSS)[4]. These applications have simplified user interaction by requiring the user to provide them with only the criteria of the channels they are interested in, instead of the complete set of channels. They have improved performance with the use of PVManager to manage the collection of control system data.

INTRODUCTION

The High level applications tend to prefer an hierarchical view of the control system name space where they can group channel by location, physical function, etc. These applications also require connections to large set of channel access connections to retrieve the value, alarm state etc.

Using the ChannelFinder we are able to provide High level applications with a hierarchical view of the control system with a simple mechanism to create muti-channel connections using PVManager.

ChannelFinder Service

ChannelFinder tries to overcome the flat name space limitation of the EPICS Channel Access protocol by implementing a generic directory service, which applications can query for a list of channels that match certain conditions, such as physical functionality or location. It also provides mechanisms to create channel name aliases, allowing for different perspectives of the same set of channel names.

ChannelFinder directory server is implemented as a REST style web service, with a relation database (RDB) backend. A client sends an HTTP request to the service to query, create, update and delete directory entries. The data (list of channels and their attributes) is sent/returned in XML (or JSON) notation.

PVManager

PVManager is client library aimed to simplify gathering and composition of control system data. When writing a client application for a control system, common elements always have to be implemented such as rate decoupling, queuing, caching, aggregation, notification on the the correct thread. PVManager allows you to assemble your pipeline starting from commonly used elements , instead of re-implementing every time.

*Work supported by U.S. Department of Energy
#shroffk@bnl.gov

PVManager provides support for composite datatypes like MultiChannelArray, MultiChannelMaps, etc. and provides mechanisms to define expressions to govern the manner in which the control system data is collected and presented to the application.

ChannelFinderClient (CFC)

An eclipse plugin which provides various clients to the various CSS applications to query the channelfinder service. The plugin also provides a set of utility methods to simplify the querying of the service and perform some common operations on the result of these queries.

ARCHITECTURE

The figure 1. shows the architecture of the new CSS applications which use the ChannelFinder service along with PVManager and CFC library plugins. The applications use the CFC plugin, to query the service for a group of channels based on set of criteria which might include positional, physical or operations properties and/or tags associated with the channels, the resulting channels are then used by PVManager to create channel access (CA) connections to retrieve their values.

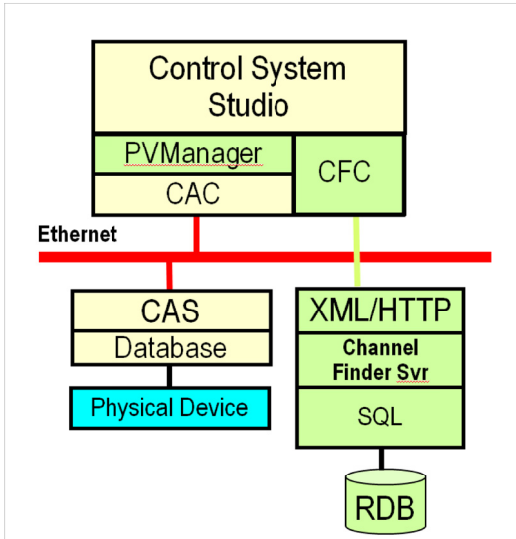


Figure 1: Architecture.

CSS APPLICATIONS

ChannelViewer

The ChannelViewer is a simple graphical application which can be used to query the channel finder. The input

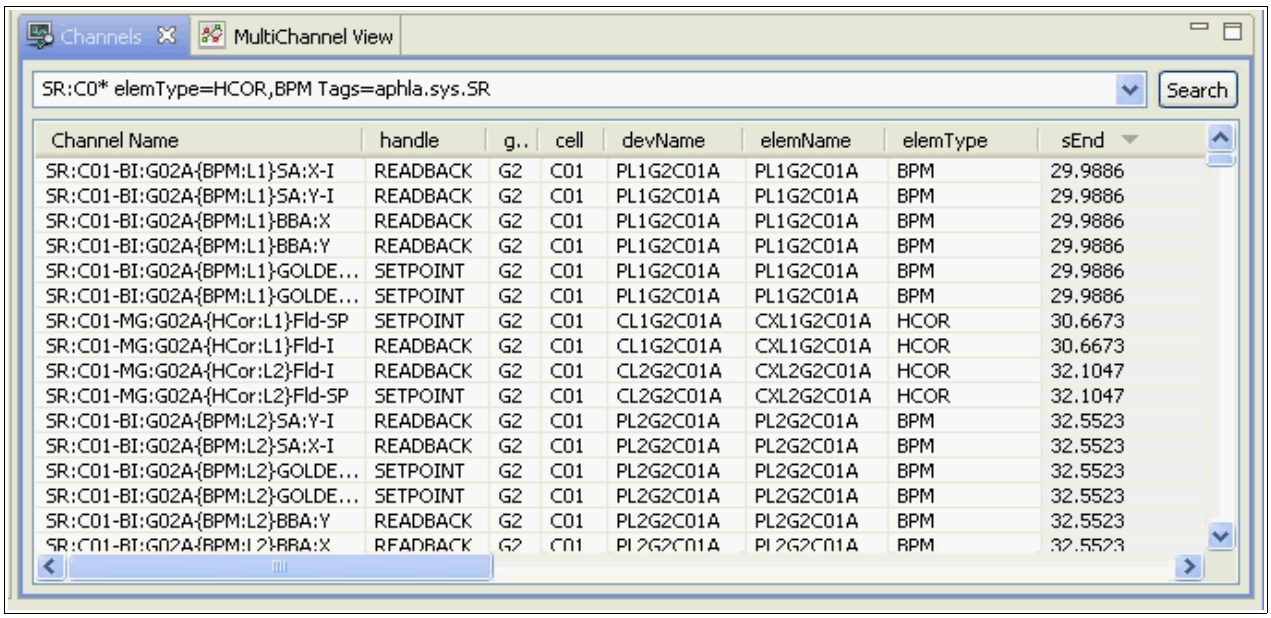


Figure 2: The Channel Viewer can be used to query the ChannelFinder service for a set of channels. The figure shows the result of a query for all the channels in the Storage ring associated with Beam Position Monitor (BPM) or Horizontal Correctors (HCOR) with the tag aphla.sys.SR.

consists of a query can be based one or any combination of the criteria: Channel name, Property values, Tags.

The resulting Channels can be sorted, grouped and tagged. They can be exported to any other CSS application as a set of process variables, thus the end user need not remember details of each process variable but can rather work with logical group determined by the query conditions.

MultiChannel Viewer

The MultiChannel Viewer application shown in Figure 3 is designed to plot the values of a group of channels which are ordered based on a user specified criteria i.e. s-position

The MultiChannel viewer accepts a query (similar to the one used in ChannelViewer) which represents a logical group of channels on the ChannelFinder server.

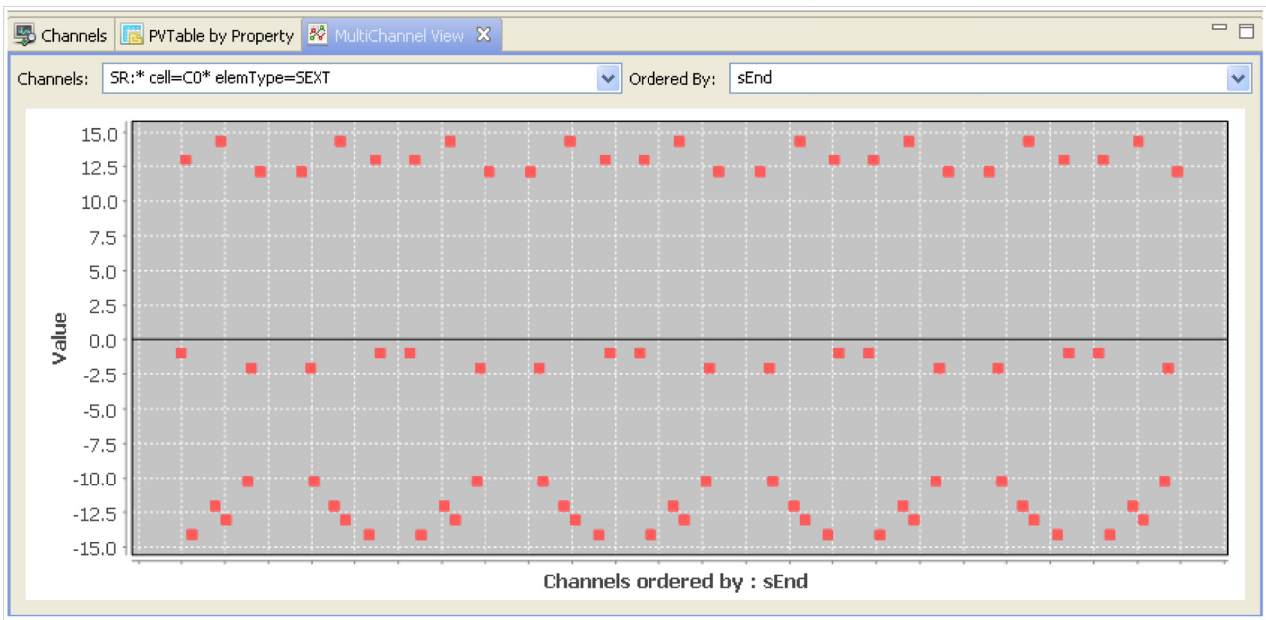
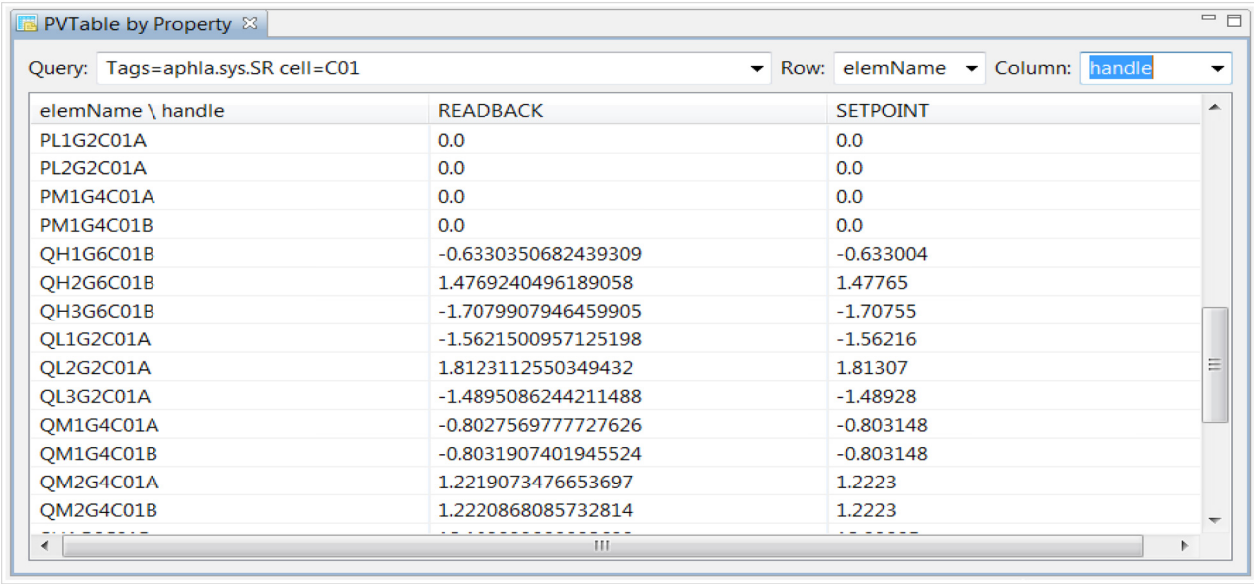


Figure 3: Shows an example use case of the MultiChannel Viewer, the application queries ChannelFinder for all the process variables associated with sextapoles(elemType=SEXT) in the storage ring (SR:*) and plots each channel ordered by the sEnd value.



elemName \ handle	READBACK	SETPOINT
PL1G2C01A	0.0	0.0
PL2G2C01A	0.0	0.0
PM1G4C01A	0.0	0.0
PM1G4C01B	0.0	0.0
QH1G6C01B	-0.6330350682439309	-0.633004
QH2G6C01B	1.4769240496189058	1.47765
QH3G6C01B	-1.7079907946459905	-1.70755
QL1G2C01A	-1.5621500957125198	-1.56216
QL2G2C01A	1.8123112550349432	1.81307
QL3G2C01A	-1.4895086244211488	-1.48928
QM1G4C01A	-0.8027569777727626	-0.803148
QM1G4C01B	-0.8031907401945524	-0.803148
QM2G4C01A	1.2219073476653697	1.2223
QM2G4C01B	1.2220868085732814	1.2223

Figure 4: Example of a full-width figure showing the distribution of problems commonly encountered during paper processing. This figure is labeled with a multi-line caption which has to be justified, rather than centered.

The result of this query, a group of channels, is then plotted with their order along the x-axis being determined by the "Order By" property selected in the drop box.

PVTable by Property

Figure 4 shows the PVTable by Property application. It allows user to organize the output of many channels in a table, based on the value of two properties, thus each cell of the table represents the current value of a single channel, the position of each channel is determined by the value of selected row and column properties. The list of channels and the properties used to create the rows and columns are obtained by querying the ChannelFinder.

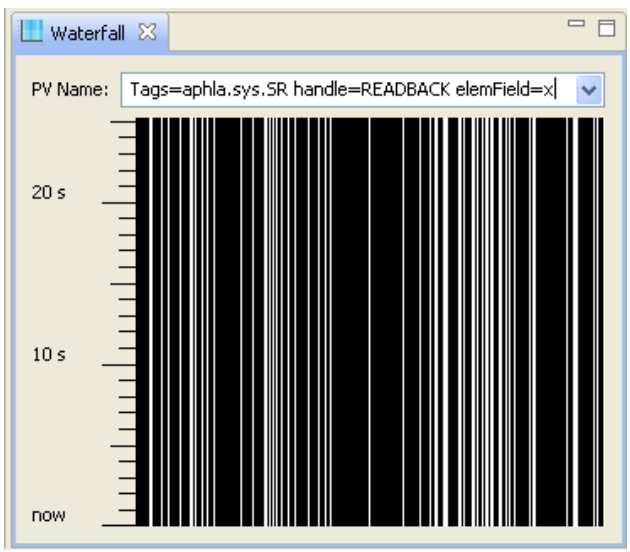


Figure 5: Waterfall plot generated from PVManager using the the result of the query for all channels with Tag=aphla.sys.SR and are readback values for element field x.

WaterFallPlot

The Waterfall plot shown in Figure 5. creates a time plot for all the values of a channel query or of a waveform.

CONCLUSION

- Use of the ChannelFinder service and client eliminates the need for configuring and remembering each individual channel.
- PVManager simplifies and improves performance of applications by handling various problems associated with use of large number of channel.

REFERENCES

[1] L. Dalesio, D. Dohan, R. Lange, A Proposal for introspection in EPICS, THP036, ICALEPS (2009).
[2] R. Lange, ChannelFinder: A directory service; http://channelfinder.sf.net.
[3] G.Carcassi, pvManager, EPICS Fall meeting (2010) ; http://pvmanager.sf.net
[4] Control System Studio; http://cs-studio.sf.net

THE SPIRAL2 RADIOFREQUENCY COMMAND CONTROL

D. Touchard, C. Berthe, P. Gillette, M. Lechartier, E. Lécorché, G. Normand GANIL, Caen, France
Y. Lussignol, D. Uriot CEA/DSM/IRFU, Saclay, France.

Abstract

Mainly for carrying out nuclear physics experiences, the SPIRAL2 facility based at Caen in France will aim to provide new radioactive rare ion or high intensity stable ion beams. The driver accelerator uses several radiofrequency systems: RFQ, rebunchers and superconducting cavities, driven by independent amplifiers and controlled by digital electronics: the low level radiofrequency subsystem which is integrated into a regulated loop driven by the control system. A test of a whole system is foreseen to define and check the computer control interface and applications. This paper describes the interfaces to the different RF equipment into the EPICS based computer control system. CSS supervision and foreseen high level tuning XAL/JAVA based applications are also considered.

SPIRAL2 AND RF CAVITIES

The SPIRAL2 project is originally designed to produce new intense rare ion beams (RIB). The main production scheme is based on fast neutron induced fission of uranium target using a carbon converter. The driver accelerator facility composed by sources, followed by a Radio Frequency Quadrupole (RFQ), 3 bunchers, and a superconducting linear accelerator (LINAC) will be able to accelerate also proton, deuteron or heavy ion beams. The LINAC will be composed of 26 quarter wave superconducting resonators closed into 19 cryo modules. All cavities will be driven by independently power amplifiers at 88.0525 MHz [1].

COMMAND CONTROL MAIN CHOICES

Two kinds of users are identified today. On one hand beam drivers and operators working in the control room are in charge of delivering beams. On the other hand, engineers have to make sub systems and equipment working even if the general control command system is not available.

Figure 1 gives an overall view of process control architecture. As usual command control systems, two main layers can be identified. The supervision layer provides high level tuning applications, supervision screens and specific human interface tools. The process control layer provides access to different actuator and feedback information generated by sensors.

For the command control room, and control process associated, EPICS was early chosen to ease development and integration of software components developed by all laboratories involved in the SPIRAL2 collaboration. Industrial PC Linux or VME Vxworks crates will host EPICS input output controllers (IOCs) [2][3].

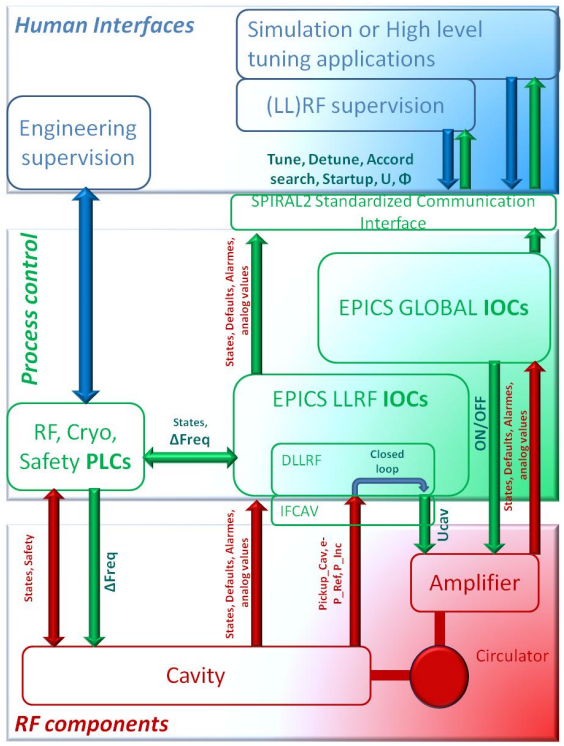


Figure 1: Main RF CC functional architecture.

Siemens S7 programmable logic controllers (PLCs) will be dedicated to slow control material protection, safety and local engineering applications. On one hand, supervision screens are produced with EDM or CSS/BOY, on the other hand more sophisticated high level tuning applications derive from the Java XAL framework [4][5].

To control Radio Frequency (RF) equipment, a dedicated command control (CC) sub system is identified. The main piece of this sub system is a low level radio frequency (LLRF) control process which will regulate the accelerating field of each cavity with a fast automatic closed loop [6].

POWER AMPLIFIERS

Amplifiers foreseen to drive RF cavities will be able to deliver up to 20 kW for solid state amplifier and up to 60kW for tube amplifiers. Both are provided by industrial companies and a special care has been taken on prototyping and testing. To ease engineering support, supervising PLCs applications are delivered with. A special care is going to be taken for putting the tests with EPICS components that will be used for the final command control [7].

Figure 2 shows the CSS/BOY supervision screen developed for bench tests of tube amplifiers.

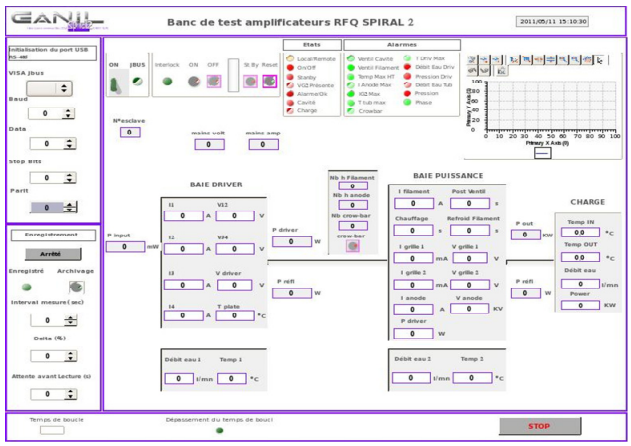


Figure 2: Tube amplifier supervision screen.

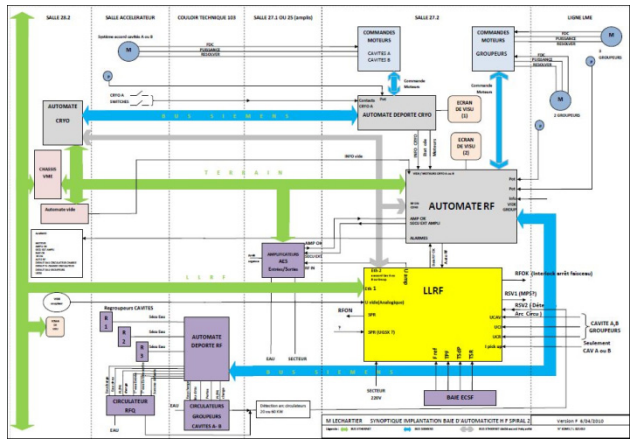


Figure 3: Implantation diagram.

LLRF CC

The LLRF will mainly provide the control analogue signal to each cavity's amplifier. Among others, it will also control the frequency, the start-up in an automated way, and will monitor the electric arc phenomenon. In order to fulfil these functionalities, the first digital subsystem (DLLRF) will acquire and process data from the second subsystem (IFCAV) which will host RF components to interface cavities. In addition to control the amplitude and phase, the DLLRF will be in charge of communicating with global command control applications throw a standardized interface. To ease the integration in the whole command control architecture, technical choices for this sub system respect the main EPICS and VME command control choices.

PROCESS CONTROL

VME Process Control

To ease high level application access to the process control layer, a SPIRAL2 standardized communication interface has been designed [8]. Dedicated RF applications will tune a cavity with the U voltage value and the Φ phase value. The EPICS channel access (CA) protocol and this standardized interface enable applications to discover equipment remote values, and lists of state, defaults, and alarms.

Global EPICS IOCs aim to interface specific RF equipment command to supervision layer. They could also be an interface to LLRF EPICS IOCs that include all specific cavity functionalities.

Implantation Diagram

Figure 3 shows implantation diagram designed for RF command-control. This implantation could evolve before final integration of each command control component, engineering needs, like cable path or command control crate, are nearly well identified today.

PLC Process Control

In addition to general functions described above, PLCs take a particular place for RF command control. Amplifiers will be interfaced via dedicated PLCs solution. In the same way, a well tuned and regulated cavity which drifts in frequency should be adjusted by mechanical or thermal constraint. As shown on figure 1, for the special case of mechanical according cavity delta frequency, dedicated brushless motors driven by PLC are under prototyping.

HUMAN INTERFACES

Introduction

To drive a machine like the SPIRAL2 facility, a set of high level and supervision applications as shown on figure 4 are going to be developed [9]. After configuring equipment, conditioning dedicated sub systems like RF equipment, and setting theoretical beam values, RF equipment can be tuned and optimized for different kind of beams. CSS/BOY applications will assume to put each equipment and servitudes into work when more complicated JAVA XAL applications will tune the machine.

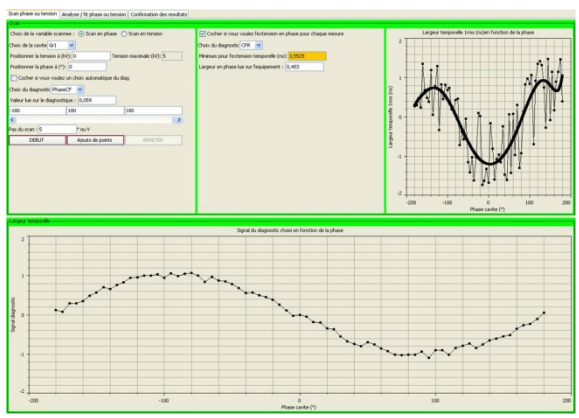


Figure 4: Cavity tuning application (preliminary).

High Level Applications

Two steps are foreseen to tune rebunchers and LINAC cavities.

At first, consigns will be set on cavities and associated quadrupole magnets, one after the other, with an application named "Cavity tuning" shown on figure 4. Theoretical values come from simulation application software named GenLinWin [10] developed by "D.URIOT et al." at CEA. Scans around theoretical amplitude and phase values associated with a time of flight measure value will be used.

Secondly the "Optimization" application shown on figure 5 will adapt the beam to the channel created, using the three rebunchers and four quadrupole magnets.

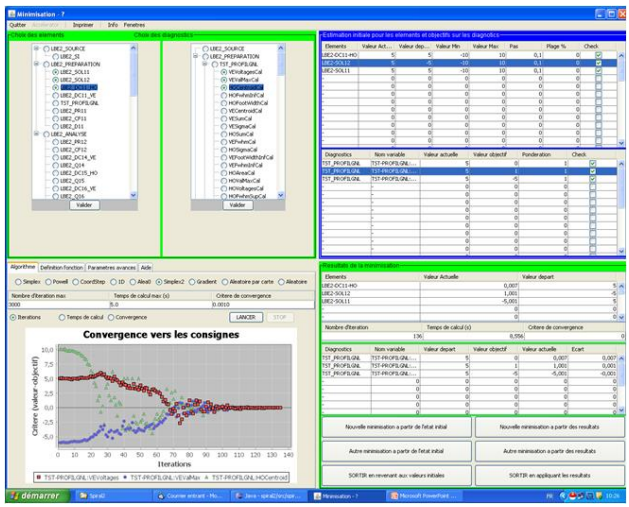


Figure 5: Optimization application.

REBUNCHER CC TESTS

Rebunchers (figure 6) used on the medium energy line have been prototyped at GANIL. Power tests started in 2010 have been performed till July 2011 [11]. They have confirmed the Q factor, the shunt impedance and maximum substained voltages as well as cooling and coarse tuning capabilities.

Figure 6: Rebuncher prototype.

The next step planned on the beginning of 2012 will qualify this cavity with the whole LLRF and CC systems. LLRF sub system has already been used on partial test-bench platforms to qualify the two families (A and B) of the LINAC cavities. Furthermore, setting and validating the whole command control during these tests will be an essential step before the final integration phase. To achieve this task, special RF CSS/BOY supervision screens to control the rebuncher will also be developed.

FIRST RESULTS AND NEXT STEPS

RF command control architecture is today well seen and designed. The tests already made on the amplifier bench have confirmed the interface between IOCs and PLCs. First LLRF tests on cavities A and B have shown that first command control prototype components reach projected performances. Simulations on the first RF tuning applications confirm the choice of XAL/JAVA. Brushless motors prototype is nearly well known now and next rebuncher tests should confirm the whole command control system. After that, the strategy to integrate the definitive command control should be confirmed.

ACKNOWLEDGMENTS

Authors would like to acknowledge warmly M. Di Giacomo J.F. Leyge for their work and availability. They take this opportunity to thank P. Baret, F. Bucaille, B. Ducoudret, G. Duteil J.C. Deroy, C.H. Patard and J.F. Roze who have made the study and the completion of these first results possible.

REFERENCES

[1] M. Di Giacomo, "Status of the RF system for the SPIRAL2 LINAC at the beginning of the construction phase", LINAC 2006, Knoxville, USA
[2] E. Lécorche, "Overview of the Spiral2 Control System Progress", ICALEPCS 2011, Grenoble, France
[3] D. Touchard, "Status of the future SPIRAL2 control system", PCaPAC 2010, Saskatoon, Saskatchewan
[4] J. Galambos, "XAL application programming framework", ICALEPCS2003, Gyeongju, Korea
[5] T. Pelaia, "XAL status", Proceedings of ICALEPCS07, Knoxville, Tennessee, USA
[6] P. Galdemard. "The development of the digital LLRF system for the SPIRAL2 accelerator" Spiral2 week2009
[7] M. Di Giacomo, "The test bench for the power amplifiers of the SPIRAL2 SC LINAC", SRF 2007, Beijing, China
[8] C. Haquin, "A Standardized Interface between High Level Applications and EPICS IOCs", ICALEPCS 2011, Grenoble, France
[9] L. Philippe, "First High-level Java Applications Based on the OPEN-XAL Library", ICALEPCS 2011, Grenoble, France
[10] Logiciels at IRFU/SACM, http://irfu.cea.fr/Sacm
[11] M. Lechartier, "RF power tests and results of the first rebuncher for the SPIRAL2 driver", LINAC2010, Tsukuba, Japan

TOOLCHAIN FOR ONLINE MODELING OF THE LHC

G.J. Müller, X. Buffat, K. Fuchsberger, M. Giovannozzi,
S. Redaelli, F. Schmidt, CERN, Geneva, Switzerland

Abstract

The control of high intensity beams in a high energy, superconducting machine with complex optics like the CERN Large Hadron Collider (LHC) is challenging not only from the design aspect but also for operation towards physics production. To support the LHC beam commissioning, efforts were devoted in the design and implementation of a software infrastructure aimed at using the computing power of the beam dynamics code MAD-X in the framework of the JAVA-based LHC control and measurement environment. Alongside interfaces to measurement data as well as to settings of the control system, the best knowledge of machine aperture and optic models is provided. In this paper, we will present the status of the toolchain and illustrate how it has been used during commissioning and operation of the LHC. Possible future implementations will be discussed.

MOTIVATION

Since the startup in 2009 the LHC has been smoothly commissioned and reached an outstanding performance. The 2011 target integrated luminosity of 1.0 fb^{-1} was already reached in June and currently increased to a total of 4.0 fb^{-1}. This was to a great extent achieved by the two-step reduction (initial 1.5m and 1.0m as of 09'2011) of β*, the extremal optical β-function in the interaction point (IP). New settings had to be generated for the LHC for each of these β* reductions. A flexible, extendable and maintainable framework was required, which allows the transfer of simulation data to the control system and verification of the generated settings which are based on this input.

The complexity of the LHC itself, combined with the required possibility to handle of a variety of complicated operational configurations, especially in the IP's, raised a demand for online modeling. There is an interest in simulations representing the current status of the machine to determine e.g. the beam position at all machine elements or the beam size according to the configuration resident in the machine. Furthermore the simulation of changes before they are driven to the machine is essential to estimate their impact concerning restrictions imposed by the aperture of the machine and machine protection.

In this paper we present the toolchain which was developed to cope with the above mentioned challenges. Applications are provided to address individual problems and a core software structure was established to provide a simulation environment used by the *Aperture Meter* [1]. The work is based on an initial implementation of a framework for the commissioning of the transfer lines [2] which was extended for the LHC commissioning [3] in 2010.

ENVIRONMENT

The main purpose of the online modeling is to establish the connection between the accelerator simulation and the real machine. The relevant systems the toolchain has to interface with, are presented in the following.

Accelerator Physics Simulation MAD-X is the de facto standard accelerator lattice design and simulation software at CERN. A large amount of machine lattice models and optics for MAD-X are available and maintained by the accelerator physics community. Alongside other features the software can calculate the optical functions and the aperture information for a given machine configuration. MAD-X input is defined via its own scripting language. The optics of an accelerator is described by a specific set of optics functions (twiss) calculated by MAD-X. The optics include the design orbit and the β-function. To calculate an optics in MAD-X, a strength value has to be defined for each lattice magnet. JMad the JAVA API to MAD-X [4] was chosen as the interface to the computing power of MAD-X. JMad provides object-oriented access to MAD-X and allows to define and maintain machine and optics definitions in `JMadModelDefinition`'s hiding the complexity of the script based interaction.

Control Infrastructure Access to the control system and online measurement data of the LHC is provided by the LHC Software Architecture (LSA) [5]. Two essential entry points are exposed to clients: The JAVA API for Parameter Control (JAPC) to retrieve measurement data and the LSA Client API for access to all LSA functions related to optics and settings. The main concepts *Parameter* (settable or measurable entity), *Context* (collection of parameter values over a period of time), *Setting* (representation of a parameter value) and *Trim* (manipulation of a number of settings) are explained in detail in [5]. Besides the online measurement data from JAPC, the CERN Accelerator Logging Service (CALS) [6] provides a JAVA API to retrieve measurement data from the logging database.

Optics (strengths and twiss outputs) have to be uploaded to the LSA database as input for the settings generation. A context (in the case of the LHC: *Beamprocess*) is defined by the sequence of optics that should be established in the machine when the context settings are driven to the hardware. In the example of the LHC squeeze all required optics are defined from the injection optics to the current 1m β* optics at the end of the squeeze beam process. The LSA settings generation is used to generate all required settings for the LHC devices. In the case of the power converter settings all parameter settings in the hierarchy are generated

from the optics strength input: PC strengths (K) down to the PC reference currents (IREF) which are driven to the hardware.

Adjustments of physics parameters like tune, chromaticity, coupling as well as orbit steering with bumps for beam crossing and separation in the IPs is done via knobs. A knob is a LSA parameter which has a unique hierarchy of dependent K parameters called knob definition. Each assigned parameter K_i is connected to the knob by an optics dependent multiplication factor f_i. Therefore a trim on the knobs setting value A is propagated to the dependent parameters like $\Delta K_i = \Delta A \cdot f_i(\text{active Optics})$.

REQUIREMENTS

The following requirements have to be fulfilled by an online modeling toolchain:

1. handling of the different machine optics configuration and data transfer to LSA
   (a) definition of optics models and tools to follow their evolution
   (b) automatic creation of optics and knobs
   (c) transfer of optics/knob data to the control system
   (d) verification of generated settings in LSA
2. provide the information of the mechanical aperture for the whole machine, including the information available from cold bore measurements and the movable device positions (e.g. collimator's and roman pots)
3. create a simulated machine, based on the current state and settings driven to the hardware
4. combine measurement data with the simulation data (e.g. orbit interpolation)
5. simulate the effect of parameter variation (virtual knobs)
6. support the extraction of all control system settings and measurements required to build a simulated model for any given time in the past

ARCHITECTURE

Package Structure Suitable for the variety of systems the online model (OM) toolchain needs to interact with, the functionality is distributed over eight 'library/service' projects and three main application projects as shown in Fig. 1.

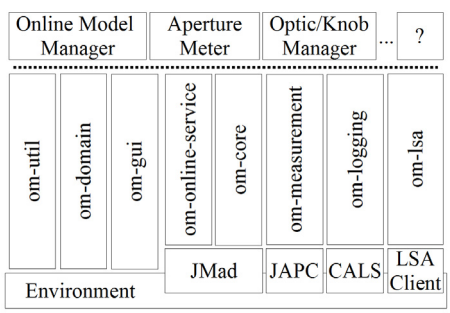


Figure 1: Online Modeling Toolchain Project Overview.

This modular design, allows to maintain and improve each part separately. The *om-util* project contains utilities for file interaction, date/time formatting, etc. The *om-domain* project contains data objects which are shared toolchain wide. Common GUI elements are gathered in *om-gui* project. The *om-online-service* project provides the online simulation from LSA settings and is explained in more detail later on. Calculations and the aperture model are implemented in the *om-core* project. The *om-measurement*, *om-logging* and *om-lsa* projects contain interfaces to interact with their respective systems, providing access to measurement or setting data. Online model applications are independent as far as possible from external packages and depend on the toolchains own packages according to the functionality required. By following this approach improvements, adjustments and bug fixes need to be performed only in one place and all application profit.

Build Project Maintaining eleven projects and ensuring that they are released in the right sequence to correctly resolve the dependencies, can be a cumbersome and time consuming task. Therefore a project was introduced into the OM framework which allows with a one click operation to release all OM packages in the correct sequence. It also provides some tools to check the currently released versions and when they have been released.

Continuous Integration and Testing To ensure that the functionality provided by the online model toolchain resumes performing in the expected manner, unperturbed by updates of software the framework depends on (JMad, LSA, ...), automatic code testing and building has been introduced. Therefore suited project/packages have been equipped with JUNIT [7] test cases covering the critical functionalities and automatic build plans have been created on the continuous integration server BAMBOO [8] provided by the CERN BE/CO group. This set-up allows to create dependency relations between the build plans of the required projects and the build plans of the OM projects. Code changes committed to the code repository trigger the building and testing of all projects related to the changed project.

FUNCTIONALITY

Optics and Knob Management

Optics Models To cope with the increasing number of optics required for simulation and operation (a total of 133 optics), it was decided to define them in `JMadModelDefinitions`. This interface is defined by JMad and is used to create a `JMadModel` which is the facade to one dedicated MAD-X process. A `JMadModelDefinition` defines the required MAD-X files to initialize the desired machine and a collection of `OpticDefinitions` which can be loaded to the `JMadModel`. An `OpticDefinition` is a named entity that

defines the required set of MAD-X input files and the sequence in which they have to be loaded in MAD-X to specify a certain machine optics. Mechanisms are provided in JMad to save the `JMadModelDefinitions` to XML files and synchronize the required input files with the optics repository, maintained by the accelerators and beam physics group. Dedicated projects have been created containing the `JMadModelDefinitions` and all required input files. These projects are in the controls software repository under version control.

When new optics are required, the corresponding `OpticDefinitions` are created and added to the proper `JMadModelDefinition`. The XML file is generated, the required input files are loaded from the optics repository into the project folder and the changes committed to the software repository. In the following build step all files are packaged into a jar file. Client application simply have a dependency to the jar which contains the required `JMadModelDefinitions`.

This set-up allows client applications to use MAD-X computation completely independently of MAD-X syntax and required input files. New optics can be used without code changes in the clients. The model definition jars represent a common source of optics definitions to ensure that simulations are performed using the same input. The versioning of the model projects allows to perform simulation on historic input data. The interaction with JMad is described more in detail in [4] and goes beyond the content of this paper.

Optic/Knob Manager Application The generation and transfer of simulation data to the control system LSA is performed using the Optic/Knob Manager. As shown in Fig. 2 it uses JMad as calculation engine and the LHC JMad model package as source for optics definitions. The available optics models are: the nominal optics model for physics production at 1m β*, the Achromatic Telescopic Squeeze (ATS) Scheme [9] optics model and the 90m β* Un-Squeeze [10] optics model for e.g. the **TOT**al cross section, **E**lastic scattering and diffraction dissociation **M**easurement (TOTEM) experiment.

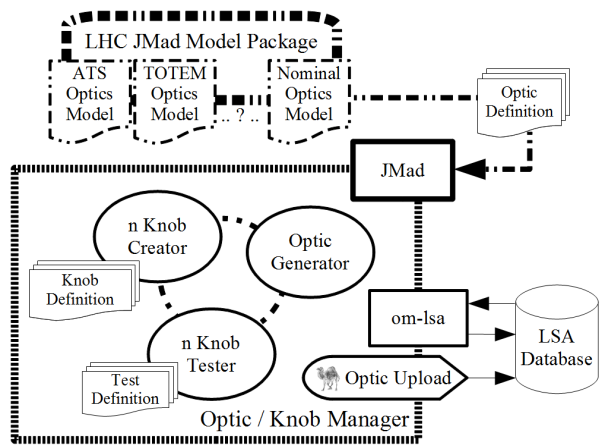


Figure 2: Optic and Knob Manager.

A simple GUI is provided in which the user can select the optics model and a set of optics to generate and upload. The `OpticGenerator` is responsible for loading the requested optics to JMad, executing possible checks for tune and chromaticity and writing the optic functions to a twiss file. A set of PERL scripts is then executed in a JAVA process to upload the generated optic data to LSA.

The same GUI also allows the selection of knobs to create for a selection of optics. The available knobs are defined by `KnobDefinitions` which have an assigned `KnobCreator` which is used to create the knob from the `JMadModel` passed. A `KnobManager` is responsible to loop over all requested optics, load the optic to a `JMadModel` and for each requested knob initialize the assigned `KnobCreator` with the model and trigger the creation process. `KnobDefinitions` are available for knobs defined by flags in the MAD-X input files (separation and crossing knobs), knobs created by MAD-X matching routines for global machine parameters like tune and chromaticity as well as for 4-corrector knobs used for local adjustment (luminosity leveling knobs). This design allows the extension of the set of available knobs by creating solely a new `KnobDefinition` and the required `KnobCreator` without touching the creation routine. `KnobTesters` are defined to ensure that newly created knobs follow definitions already present in LSA, are created correctly or produce the desired setting change when trimmed in simulation.

The import of knobs defined as a set of name value pairs is supported to define arbitrary knobs in LSA. Names can be one of the following: LSA parameter names, LSA device names or MAD-X strength names. The `om-lsa` package is finally used to transfer knob data to the control system as well as retrieve available knob definitions from the LSA database. The possibility to copy existing knob definitions to other optics is provided as well.

Setting Verification

To ensure that the settings generated in LSA are complete and correct and to apply possible corrections, the following tools have been developed in the OM toolchain based on the extraction functionality provided in the `om-lsa` project.

Beamprocess Scanner This tool can scan over the full power converter setting functions defined in a beamprocess. A snap shot of the settings is extracted at given times in the setting functions of a beam process. The settings are translated to MAD-X strength values and loaded in JMad to calculate the optic functions. The key optics parameters (tune, chromaticity, beta-beat...) can then be evaluated as function over the beam process duration. The tool has been essential for the setting optimization for the LHC squeeze [11]. It can be used to verify that the settings are valid in general (e.g. they produce reasonable tune and chromaticity evolutions) and to precalculate feed-forward correction for tune and chromaticity [12].

Online Model Manager Next to other functions, it is possible to visualize the sequence of optics in a beamprocess together with the nominal evolution of the optics key parameter values (tune, chromaticity and β*). The display acts as selection interface for the time in the beamprocess where the power converter settings should be extracted, either to display or to write them to a file in MAD-X input format. When loaded to display, the generated settings can be checked with respect to the nominal settings expected from the optics input saved in LSA. The settings saved to file can be loaded in a MAD-X script to calculate and display the optic functions. A plotting environment is provided that allows the display of multiple e.g. orbits on top of each other. The evolution of knob bump shapes can be visualized by a sequence of knob trims performed with the LSA Trim application, followed by an extraction and a MAD-X run based on the extracted settings.

Online Aperture Model

The *static* aperture model is implemented in the `om-core` package. It provides the dimension of the clearance in the vacuum layout in transverse coordinates. The data is based on the theoretical aperture with the cold bore measurement results. The model was extended with the current positions of the movable devices (collimator's and roman pots), read online from the hardware. This functionality is provided by the `om-measurement` package. The measured data from the movable devices are translated into aperture information by using the movable device information (angle, type, beam it acts on,...) that are available from the `om-lsa` package. The aperture model is an essential ingredient for the aperture meter [1].

JMad Online Service

The `JMadOnlineService` is implemented in the `om-online-service` package as the online extension to JMad. The current configuration of the machine is simulated based on the active optic and the optic dependent definitions and current settings of a set of knobs that represent the relevant orbit configuration (separation, crossing, luminosity leveling knob settings and a set of aperture scan knobs). The active optic is the one that was the basis for the generation of the settings that are currently resident in the machine hardware. A `ModelManager` allows to access, manipulate and retrieve data from the underlying `JMadModels`. Two manipulators are provided to transfer settings from LSA to the underlying `JMadModels`: One can update the models to the current state resident in the machine. The second on loads the settings that have been resident at a given point in time. This service is the basis for the aperture meter [1].

Measurement Data Treatment

Regarding measurement data the online modeling toolchain provides two more features: The interpolation of the measured orbit to all machine elements and the determination of the current state of the machine based on online data from the collimator hardware and from data distributed over the timing system. The orbit interpolation is implemented in the JMad core package and uses transfer matrices to calculate the orbit positions between two adjacent monitors. The measured orbit data are either received by a JAPC subscription to the orbit feedback service unit, or by extraction from the logging database. The identification of the online machine state is performed by merging the status and current time in the setting function retrieved from a given collimator front-end with the information about the beam mode distributed over the LHC timing system.

CONCLUSION AND OUTLOOK

The basic architecture and features of the online modeling toolchain have been presented. It was successfully used for the preparation of all LHC optics configuration for physics production as well as machine development studies and provides a good basis for future developments. The usability was shown in applications presented in this paper and as basis for the aperture meter [1]. Further developments will include the full integration of new features provided by LSA and the toolchain itself and changes towards a strict distribution of the functionality into the packages of the toolchain architecture.

ACKNOWLEDGMENTS

The authors like to thank M. Lamont, M. Strzelczyk, R. Tomas and J. Wenninger for fruitful discussion and useful advice. This work was supported by the Wolfgang-Gentner-Programme of the Bundesministerium für Bildung und Forschung (BMBF).

REFERENCES

[1] G. Müller et al., *Aperture Meter for the Large Hadron Collider*, these proceedings.
[2] F. Schmidt et al., *LHC On-Line Modeling*, Proceedings of PAC'07, 2007.
[3] G. Müller et al., *The Online model for the Large Hadron Collider*, Proceedings of IPAC'10, 2010.
[4] K. Fuchsberger et al., *Status of JMad, the* JAVA*-API for* MAD-X, Proceedings of IPAC'11, 2011.
[5] M. Lamont et al., *LHC Era Core Control Application Software*,Proceedings of ICALEPCS'05, 2005.
[6] C. Roderick et al., *The CERN Accelerator Measurement Database: On the Road to Federation*, these proceedings.
[7] JUNIT, `junit.sourceforge.net`.
[8] BAMBOO, `atlassian.com/software/bamboo`.
[9] S. Fartoukh, *An Achromatic Telescopic Squeezing (ATS) Scheme For The LHC Upgrade*, Proceedings of IPAC'11, 2011.
[10] S. Cavalier et al., *90m Optics Commissioning*,Proceedings of IPAC'11, 2011.
[11] X. Buffat et al., *Simulation of linear beam parameters to minimize the duration of the squeeze at the LHC*, Proceedings of IPAC'11, 2011.
[12] M. Pereira et al., *Feed-Forward in the LHC*, these proceedings.

CONTROLLING AND MONITORING THE DATA FLOW OF THE LHCb READ-OUT AND DAQ NETWORK

Rainer Schwemmer, C. Gaspar, N. Neufeld, D. Svantesson, CERN, Geneva, Switzerland

Abstract

The LHCb read-out uses a set of 320 FPGA based boards as interface between the on-detector hardware and the GBE DAQ network. The boards are the logical Level 1 (L1) read-out electronics and aggregate the experiment's raw data into event fragments that are sent to the DAQ network. To control the many parameters of the read-out boards, an embedded PC is included on each board, connecting to the boards ICs and FPGAs. The data from the L1 boards is sent through an aggregation network into the High Level Trigger farm. The farm comprises approximately 1500 PCs which at first assemble the fragments from the L1 boards and then do a partial reconstruction and selection of the events. In total there are approximately 3500 network connections. Data is pushed through the network and there is no mechanism for resending packets. Loss of data on a small scale is acceptable but care has to be taken to avoid data loss if possible. To monitor and debug losses, different probes are inserted throughout the entire read-out chain to count fragments, packets and their rates at different positions. To keep uniformity throughout the experiment, all control software was developed using the common SCADA software, PVSS, with the JCOP framework as base. The presentation will focus on the low level controls interface developed for the L1 boards and the networking probes, as well as the integration of the high level user interfaces into PVSS.

LHCB READ-OUT CHAIN

The LHCb DAQ system is built around an Ethernet based network, which transports the experiment's data from the detector to a Linux based filter farm. This farm consists of 1500 computers and processes the data of every accepted event and then decides whether the event is interesting or if it should be discarded. Accepted events are then sent on via IP over Ethernet to a group of computers which replicates the stream of accepted data to the disk writing processes and to a dedicated monitoring farm.

Data is accepted at a maximum rate of 1.1 MHz which translates to a data rate of approximately 50 GByte/s on the input of the network. On the border between network and detector is a set of 350 FPGA based converter boards – Trigger Electronics and Level 1 board (TELL1) [1] – which receive the data via optical links from the detector. The data is first checked for consistency and then compressed into a sub-detector specific format. After that, the event data is combined with meta information and a unique event ID. In the last step, data is coalesced into IP packets and sent to an on-board, 4 port Gigabit Ethernet (GbE) card. The TELL1s then send the data to the filter farm through two main data switches and a set of 50 fan-out switches. The total packet rate in the network is about 35 MHz.

Inside a filter farm node, the event fragments are assembled into complete events and then handed off to trigger applications which do a partial reconstruction and filter process. From the filter farm onward, the data rate drops to a more manageable rate of 200 MB/s and is again aggregated on two dedicated machines before being replicated to the online monitoring farm and the storage layer. The local storage system is only a temporary buffer for the data though. Data is written in sets of 3 GB files. Once a file is full, its consistency is verified, check sums are calculated and the file is sent off to permanent tape storage at CERN.

Throughout this chain, there are many points, at which data can be sporadically lost or which can halt the data taking process completely.

TYPICAL LOSS POINTS AND BOTTLENECKS

Read-out Boards

When a read-out board fails to produce data, there will not be enough fragments to rebuild the event in the farm and the event will be discarded. If a TELL1 stops sending any data at all, the whole read-out chain is stopped. If it cannot cope with the data rate for some reason, and slows down, the whole DAQ process is affected. In order to quickly diagnose problems inside the read-out board, the data flow is already monitored on its way through the board.

There are five FPGAs on each board, where four are responsible for checking and compressing the sub-detector data and one is assembling the data from the four preprocessors. Additional meta information from the Timing and Fast Control (TFC) system is then added to each event, before it is sent to a buffer for network transmission.

Typical problems here can be a front-end not sending data or the data transmission over the board between chips. The compression and gathering algorithms, if they are overwhelmed by too much data due to noisy channels in the detector or unexpectedly high occupancy events can also slow down the process or fail completely.

Network

At the design time of the read-out system it was deemed too unpractical to have a reliable transport protocol between the TELL1 boards and the filter farm. The huge number of farm nodes and TELL1 boards would necessi-

tate a substantial amount of memory for state information and data buffering on the read-out boards, which at that time was just not an option. The chosen protocol is a pure push protocol, where data is being sent as IP packets whenever bandwidth is available on the read-out boards. In a way the protocol behaves like UDP. In case the switches decide to drop packets, there is no retransmission.

The situation is further complicated by the fact, that all the fragments for a particular event have to be sent to a single farm node. The fragments leave the TELL1 boards all at the same time and for a short time there is a strong overcommitment of the output port to the farm node where the data is supposed to go to. The average rate to a single farm node is still far below the 1 Gigabit/s limit, but for a switch this is a typical situation to throw away packets, if it runs out of buffer space.

Farm Nodes

Inside the farm computers, the data is received by the network card, assembled into events and then passed on to the trigger applications[4]. If the machine is too busy, data can be lost already on it's way from the network card to the kernel, before it enters any application. Other problems can be trigger processes getting stuck and not processing events any more, or a machine in general having hardware problems which influences performance.

Storage Layer

As the data rate is reduced to about 200 MB/s after the trigger farm and there is only inter-computer communication, the read-out chain uses the TCP/IP protocol from the farm onward. TCP is a reliable protocol, so data loss in the network is not as much of a problem any more as in the steps before. However, switches can fail and processes can die unexpectedly, so this part of the system is monitored as well.

The actual process of writing at a sustained rate of 200 MB/s for several hours per day also requires special hardware, especially if it has to be protected from single mode failures. For data storage, we use a fiber channel based Storage Area Network (SAN) consisting of several head nodes and a fully redundant hard-disk array. The entire SAN is set up in an active-active configuration and again each connection point in the fiber network is monitored to make sure that all components continue working and share the load.

READ-OUT CHAIN PROBES

In the end one wants to be able to diagnose, as quickly as possible, why the system's performance is degraded and which link in the read-out chain is responsible. In order to find the culprit, we have – over time – inserted many probes to gather statistics throughout the entire DAQ process. At the same time, care had to be taken to not overload the components with probes, because they are usually all highly utilized by the DAQ process.

Read-out Boards

On the TELL1 boards, the number of event fragments that come in on every optical link is already counted, to find misbehaving front-ends. After the pre-processing stage, event numbers are counted as they leave the pre-processing FPGAs and when they enter the aggregation FPGA. A final count is done when they are assembled into IP packets and how many packets are sent to the GbE card.

The TELL1 comes with an embedded 486 CPU that sits on each board and which is capable of talking to all the FPGAs and other micro-chips on the board via PCI and I^2C buses. A program on the PC periodically gathers all the counters and publishes them through a unified messaging layer, called DIM[2], to the Experiment Control System (ECS) [3].

Network

Since we are using only off-the-shelf network equipment we can use the standard counters that come with the switches and routers. What makes the situation a bit more complicated is the fact that a switch does not know anything about event fragments but only about network packets. It also has persistent counters, that are not reset on every run change. Additionally, on the core routers, which have to cope with the full 50 GB/s, 35 MHz packet rate, we cannot permanently run the full statistics gathering, because it overloads their CPUs and slows down normal network operations.

Nevertheless, this is the only semi-blind spot in the system, and by monitoring the data leaving the TELL1s as network packets and the data entering the farm nodes also as packets, we can trace back problems to the switches and then check in more detail where the problem is by switching on specific counting features in the switches.

General data, like port counters are directly gathered via scripts from the ECS. For more in depth debugging sessions, the system administrators have to directly log in to the switches and check the statistics manually or use dedicated console scripts.

Farm Nodes

We also had trouble with data being lost on the way from the network card to the application. We are using the counting features of the Linux Firewall to count packets as they are arriving. The packets are filtered by source IP addresses, which can then be mapped to the sending TELL1s.

Further up in the chain we count again the number of assembled event fragments and from which source they came. Finally, the total processed and accepted/rejected events are counted as well.

Again, like on the read-out boards, a special program was developed to gather the statistical counters of the fire-

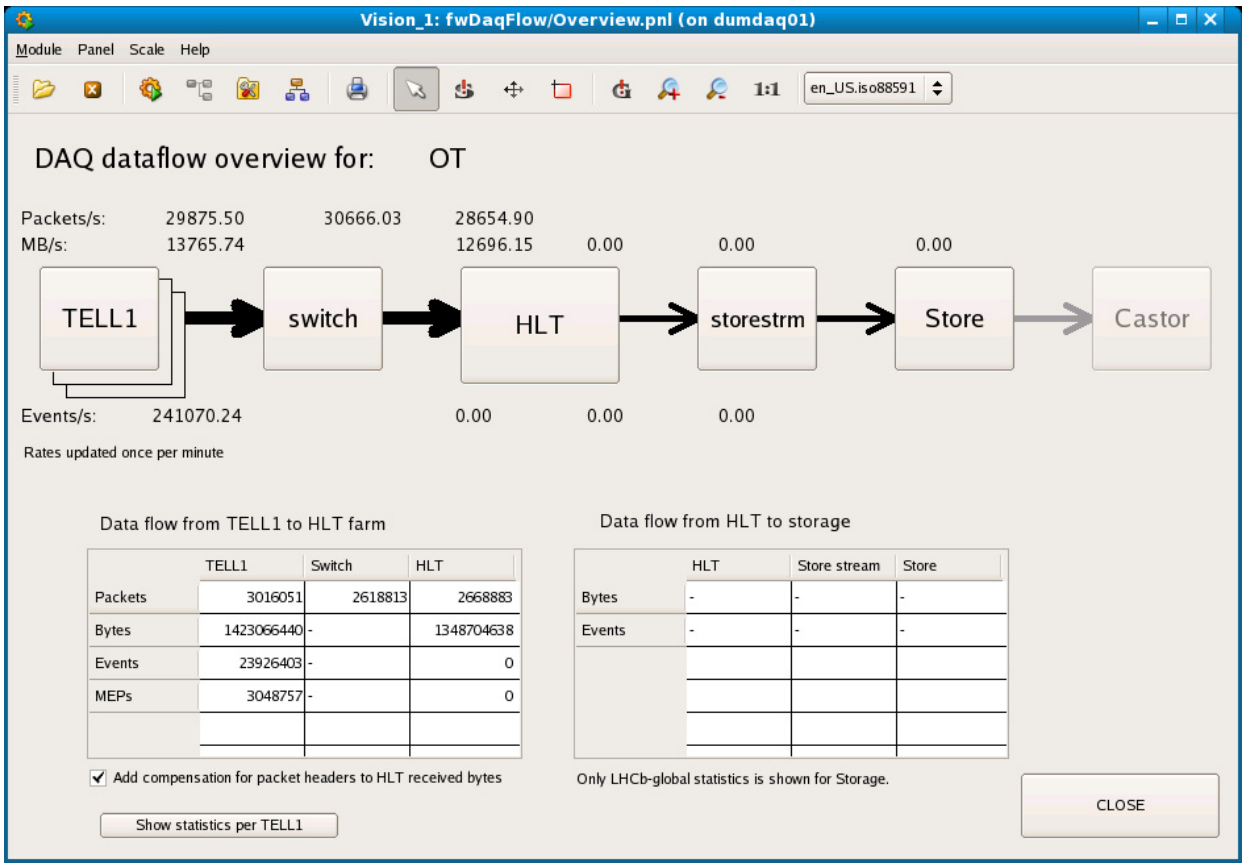


Figure 1: Top overview panel of the DAQ flow monitor. The User can click on the buttons of the different routing points to get a more a more detailed view of the subsystem.

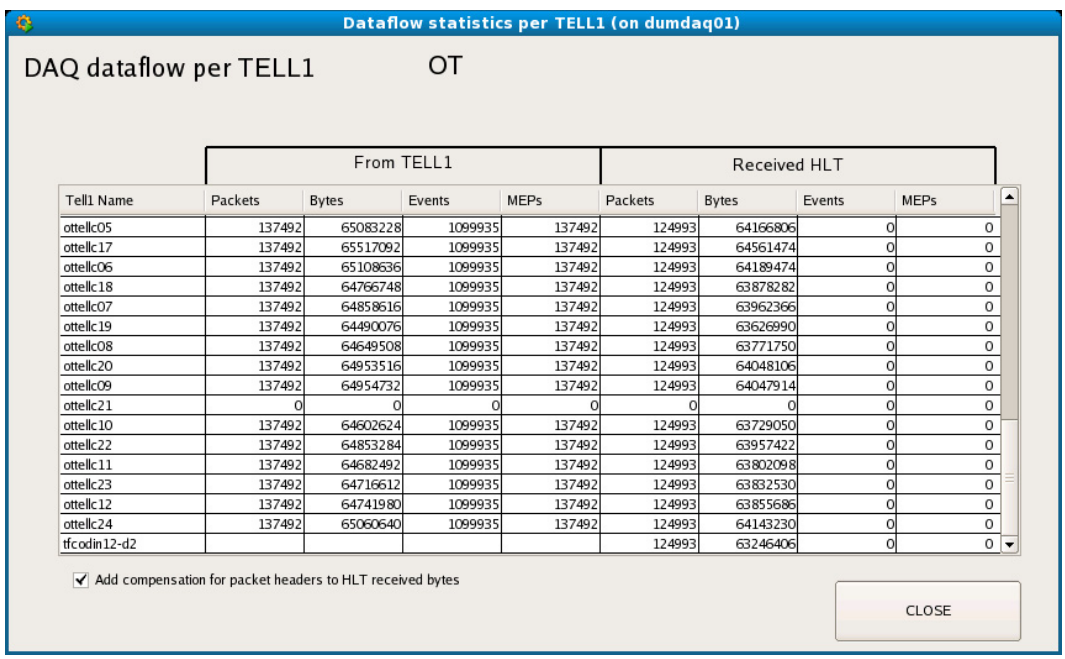


Figure 2: Detailed counters of the intersection between TELL1s and the trigger farm. ottellc21 is not producing any data and subsequently no events are produced.

wall and the trigger processes and publish them to the ECS. In this case more effort had to be put into the program, since it has to process several times the number of fragments times the number of TELL1s times the number of farm nodes.

We developed a tiered process hierarchy, where programs on individual farm nodes send their statistical information to higher level nodes where the data is added together and then sent to a central processing program, which again adds the sub-results and publishes them. The gathering programs also tap into the control system to detect when runs start and stop, and to detect which parts of the detector are actually part of the read-out. This is necessary, because the detector can be split into its individual sub-detectors which can run in standalone mode.

Storage Layer

Finally the accepted data has to be written to disk. Again there are a number of critical processes, which can fail and contribute to the loss of data. Since this part of the read-out is based on TCP, data loss is not caused by network packet drops or disappearing events, but through loss of performance. If the data cannot be written to the disks fast enough, or there are general hardware problems on the way to the disks, the system will slow down, decreasing the amount of physics data that is gathered over time.

Data arrives from the farm on two dedicated machines, which aggregates the events from the 1500 sources into file streams. There are usually several files being written in parallel for better load balancing. The files are then written by a simplified distributed file system [5]. All written events are again counted by the writing daemons and published to the ECS.

While we are still counting events up to the point where the data is written into actual files, we switch to purely rate based monitoring on the actual SAN back-end. There are fairly good standard tools like Munin[6] and problems in this area are outside the scope of what the operators of the experiment are usually dealing with. Subsequently we are using these tools directly instead of relaying the information to the ECS. Information is gathered from the storage head nodes directly, on the Fiber Channel switches and from the disk storage system itself. We developed dedicated plug-ins for munin, to monitor individual raid sets in the SAN and the throughput through each individual fiber channel path that leads to them. The results are then displayed on a dedicated web page. A permanent screen in the control room displays a sub-set of the most important rates to the operators.

USER INTERFACE

All the information gathered from the different counters and probes throughout the network has to be compiled into an easily understandable interface, which allows the user to quickly pinpoint problems. To hide the fact that there are hundreds of thousands of counters, we developed a dedicated master process, which gathers all the counters, sorts them into categories of either event counters, network packet counters or byte counters and calculates the rates of these for all the major routing points in the system.

A UI panel then shows a schematic representation of these points and the rates at which data is passing through them. In case of problems, the operator can quickly diagnose the point at which the disruption is happening. For more in-depth analysis, all the routing points have dedicated UI expert panels, which show the full multitude of counters. The top overview panel is shown in Fig. 1. In the case shown here, there is a problem with the assembly of the event fragments. Packets are arriving at the farm, but no events are produced by the event building processes. Looking at the detailed view of the TELL1 - HLT connection in Fig. 2, one can see a single TELL1 not producing any data.

CONCLUSION

We have developed a multi-tiered hierarchy of programs, which are capable of reading the various statistics counters that are present throughout the LHCb DAQ chain and moreover, are capable of coping with the enormous amount of information. The data is brought into a unified format which makes it easier to spot points of data loss in the system.

All the counters are analyzed by a dedicated process and then presented to the user in a simple panel, which hides most of the complexity of the system, but gives access to more detailed information if necessary. This system has helped us to quickly find problematic network links, failing read-out boards and front-ends and has improved the response time from the time a problem occurs to being fixed.

REFERENCES

[1] Guido Haefeli et al., “TELL1 Specification for a common read out board for LHCb”, IPHE Note 2003-02, LHCb Note 2003-007.

[2] C. Gaspar et al., “DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication”, Proc. of CHEP 2000, Padova.

[3] C. Gaspar, B. Franek, R. Jacobsson, S. Morlini, N. Neufeld, and P. Vannerem, “An integrated experiment control system, architecture, and benefits: The LHCb approach”, IEEE Trans. Nucl. Sci., vol. 51, no. 1, pp. 513520, Jun. 2004.

[4] G. Barrand et al., “GAUDI - A software architecture and framework for building LHCb data processing applications”, Proc. of CHEP 2000, Padova.

[5] J. Garnier, N. Neufeld, and S.S. Cherukuwada, “Non-POSIX File System for LHCb Online Event Handling”, Real Time Conference (RT), 2010 17th IEEE-NPSS, p 1-4.

[6] Munin: “A networked resource monitoring tool that can help analyze resource trends”, http://munin-monitoring.org/

INTEGRATING CONTROLS FRAMEWORKS: CONTROL SYSTEMS FOR NA62 LAV DETECTOR TEST BEAMS

O. Holme, CERN, Geneva, Switzerland and ETH Zurich, Switzerland
J. Arroyo Garcia, P. Golonka, M. Gonzalez-Berges, H. Milcent, CERN, Geneva, Switzerland

Abstract

The detector control system for the NA62 experiment at CERN, to be ready for physics data-taking in 2014, is going to be built based on control technologies recommended by the CERN Engineering group. A rich portfolio of the technologies is planned to be showcased and deployed in the final application, and synergy between them is needed. In particular two approaches to building controls application need to play in harmony: the use of the high-level application framework called UNICOS, and a bottom-up approach of development based on the components of the JCOP Framework. The aim of combining the features provided by the two frameworks is to avoid duplication of functionality and minimize the maintenance and development effort for future controls applications. In the paper the result of the integration efforts obtained so far are presented; namely the control applications developed for beam-testing of NA62 detector prototypes. Even though the delivered applications are simple, significant conceptual and development work was required to bring about the smooth inter-play between the two frameworks, while assuring the possibility of unleashing their full power. A discussion of current open issues is presented, including the viability of the approach for larger-scale applications of high complexity, such as the complete detector control system for the NA62 detector.

INTRODUCTION

NA62 is a fixed-target experiment [1] being constructed at CERN that aims at studying ultra-rare decays of K particles (kaons) and thus follow the research performed by the NA48 detector. The detector is scheduled to start Physics Data-taking in 2014, yet many of its components will already take part in preliminary operations in autumn 2012 to measure their performance.

The NA62 apparatus is composed of 10 subdetectors positioned inside and at either end of a 115 meter long vacuum tank. It is expected to observe around 100 ultra-rare K particle decays in two years of operation with proton beams delivered by the SPS accelerator at CERN.

The detector hardware of the NA62 experiment is a mixture of legacy NA48 equipment, such as its unique high-resolution Liquid Krypton calorimeter and brand new equipment that exploit the newest technologies, such as the ultra-fast Gigatracker [2] and the Large-Angle Photon Veto (LAV) subdetector.

In terms of controls, the detector is bigger and more complex than other contemporary fixed-target experiments at CERN, yet smaller than the scale of the LHC experiments. In the order of 5,000 high voltage channels and 10,000 analog and digital values and parameters need to be commanded and monitored.

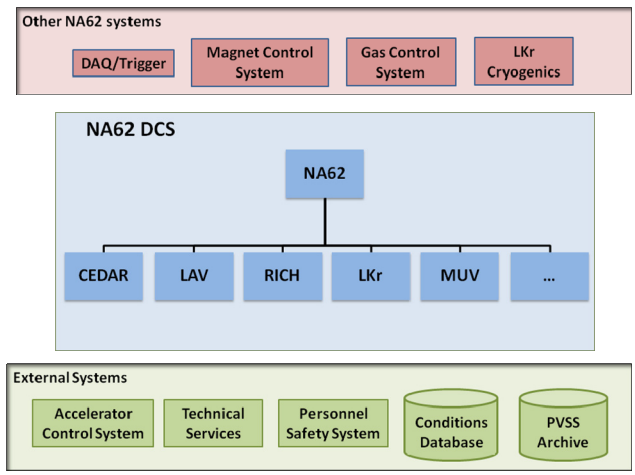


Figure 1: The NA62 DCS subdetectors and other essential systems from the CERN and NA62 infrastructure. [3]

To develop the Detector Control System (DCS) for NA62 (Fig. 1), a collaboration between NA62 and the CERN Industrial Controls & Electronics group (EN/ICE) was established in 2009. The aim is to develop and deliver the full-scale control systems for the final physics run, as well as other control systems that might be necessary for NA62 to test and commission their equipment. In this paper an example of the control systems used for beam-testing of the prototypes of the LAV subdetector is presented. These beam tests have been conducted in two-week periods of October 2009 and February and August 2010, and made use of the controls tools that are described in later sections. In April 2011 an application that monitors the environment of the storage area for the LAV equipment was developed using the same technology. This control system is still in operation.

CONTROL SYSTEM FRAMEWORKS

The supervisory control software called WinCC Open Architecture (WinCC OA is the new name for the product previously known as PVSS) [4] was selected as the basis of the NA62 DCS, in line with the selection made by the JCOP (Joint COntrols Project) collaboration for the LHC experiments. The EN/ICE group at CERN develops and supports two controls frameworks based on WinCC OA. These are the UNICOS (UNified Industrial COntrol System) and JCOP frameworks. These frameworks have been successfully deployed in large scale control systems

in the LHC accelerator and experiments, enabling considerable practical experience to be gained, permitting continual improvements and extensions of the features of both frameworks.

In addition to the WinCC OA frameworks, a suite of CERN communication systems are available for the exchange of information between systems and between the supervisory level and front-end. These systems include CMW (Controls MiddleWare) [5], DIM (Distributed Information Management) [6] and DIP (Data Interchange Protocol) [7].

UNICOS Framework

UNICOS [8] is a CERN framework developed to produce control applications for three-layer control systems (Fig. 2). UNICOS provides developers with the means to rapidly configure and generate full control or monitoring applications and provides operators with ways to interact with all the items of the process from the most simple (e.g. I/O channels) to high level compound devices with little effort.

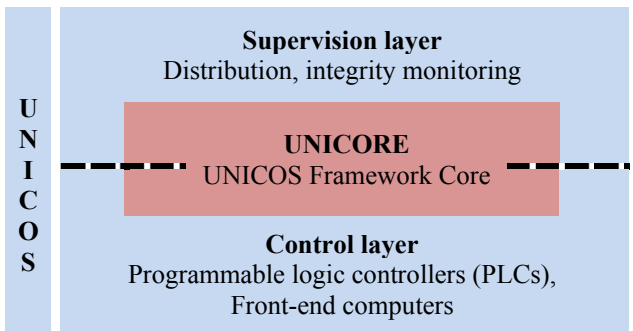


Figure 2: The three-layer control system architecture of the UNICOS Framework.

UNICOS offers tools to diagnose problems in the process being controlled and the control system itself as well as tools to access and operate the devices without specific development. The framework proposes a reusable environment composed of a set of components for the UNICOS-WinCC OA and UNICOS-Front End parts. At the supervision level in both Linux and Windows operating system, functionality such as application distribution across many SCADA data servers, monitoring the integrity of the application (e.g. status of the front-end) and configurable device trending are available. In addition it offers operators and application developers a configurable and homogenous user interface that is entirely customizable and which does not imply prior knowledge of the WinCC OA scripting language and the use of basic WinCC OA device access interfaces.

JCOP Framework

The JCOP Framework [9] focuses on providing a set of guidelines and software tools that can be used by control system developers to facilitate the building of controls applications. The framework provides a simple and standardised high-level programming interface to common functions that are required at configuration and runtime in a controls application. In addition, developers are able to directly access the underlying WinCC OA features, enabling developers to go beyond the scope of the JCOP framework in order to take advantage of all the functionality offered by WinCC OA itself.

The software tools provided allow simple creation, configuration and hierarchical organisation of instances of device classes. The device centric nature of the JCOP framework defines the architecture of the final application. The JCOP framework provides a substantial library of functionality targeted to applications based on this architecture. A pre-defined set of hardware devices is provided as standard, covering devices that are commonly used in the LHC experiments at CERN.

Functionality of the JCOP framework includes:

- Saving and restoring hardware configurations and settings, with data being stored in a database.
- Organising devices into multiple hierarchical tree views, based on off-detector and on-detector layouts.
- Control of all devices via a hierarchy with integrated Finite State Machine logic.

Comparison of Frameworks

Historically, features of both the JCOP and UNICOS frameworks were developed by coordinated effort between the relevant development teams, meaning that at a low level, several fundamental principles of the framework internals are similar. However, due to their very different aims, there are considerable differences in some conceptual models and design assumptions upon which the frameworks are based. For instance, both frameworks are based around a method of modelling classes of devices. Yet the modelling concept of each framework is different, so any integration of the two must first address the issue of how to enable interoperability of tools previously designed to work with only one of these specific model types.

UNICOS offers a methodology to create a ready-to-use controls application by following a series of configuration and generation steps. It provides a framework that encompasses the hardware interfaces up to the final operator's view of the system. This high-level, comprehensive approach allows a developer to create a complete system without needing to understand many of the internal workings of the framework or WinCC OA.

The JCOP philosophy offers a set of tools and a defined architecture which the developers of control systems can use as a basis for building the complete application. The advantage of this is that an application developer has a greater degree of flexibility in the design of the final application, particularly at the level of the operator user interfaces. However, the JCOP approach requires a good understanding of the functionality of the framework itself and more knowledge of the WinCC OA scripting language as the developer must build their application from a library of pre-defined components.

The concept of combining both frameworks in the DCS of NA62 was inspired by the different and

complementary features of the two frameworks. By combining the functionality associated with many pre-existing JCOP device classes, with the simple and holistic application generation approach of UNICOS, the maximum benefit could be derived from these frameworks. NA62 is the first target of such a unification effort, acting as a driver for the innovative work that is required for this task. It is important that the functionality should be merged seamlessly for the final application builders and operators. There is a considerable amount of design, prototyping and testing effort needed to achieve the goal of a smooth harmonisation of these frameworks.

INTEGRATION APPROACH

As mentioned previously, the device modelling concepts of UNICOS and JCOP are different and the unification of frameworks at this level is the key to interoperability of tools from each framework. Previously, the high-level UNICOS application tools could only work with UNICOS-specific devices and front-end devices. The JCOP devices cannot make use of these tools by default.

One solution to this would have been the creation of duplicate models and instances of devices for both frameworks. This approach would have increase complexity and maintenance effort and, as such, is clearly not desirable. In order to avoid such duplication, a novel proxy interface was designed (Fig. 3). This proxy enabled JCOP device instances to be visible within the UNICOS domain with little effort for the application builder.

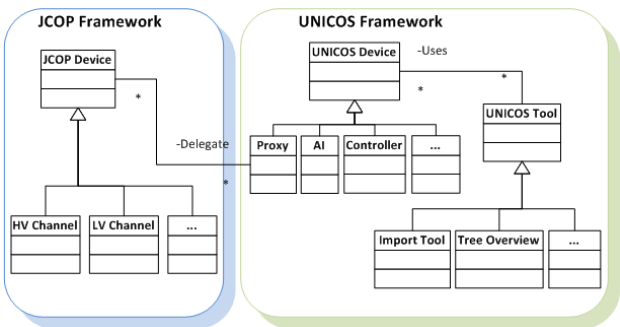


Figure 3: The proxy concept for enabling interoperability between UNICOS and JCOP domains.

In order to use this proxy mechanism, it is necessary to create a device or front-end in the UNICOS domain that contains no data. This object will act as the proxy to the JCOP device. Whenever a UNICOS utility is used with a UNICOS device, the proxy checks if the JCOP object is available and, if so, provides a transparent access to the data in the JCOP domain.

The proxy interface allows flexibility in the way devices are mapped between the JCOP and UNICOS frameworks. For instance, a UNICOS powering device could act as a proxy for a single JCOP powering channel, or a group of devices such as a multi-channel power supply.

FIRST PRODUCTION SYSTEM

As part of the collaboration between EN/ICE and the NA62 experiment, two prototypes were developed with the dual purpose of testing the framework integration concepts and providing the necessary control applications for the LAV subdetector.

The LAV subdetector is composed of 12 independent stations situated at points between 120 and 240 meters along the experiment. Each station is composed of 160 to 256 lead glass crystals with photomultipliers attached. These are assembled into rings, with 4 or 5 rings in each station (Fig. 4).

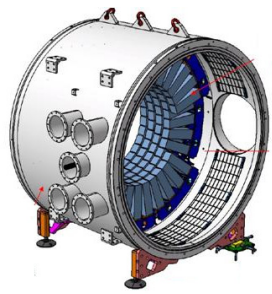


Figure 4: A representation of a LAV station. [1]

In the first LAV station alone, there are 30 PT1000 sensors to measure temperature as the LAV is sensitive to rises in temperature in the order of tenths of a degree. It is expected that 2496 high voltage channels will be needed, distributed between the 12 LAV stations.

LAV Prototype

The DCS for the LAV prototype (Fig. 5) was needed for operations of the LAV in test beams. It controlled and monitored two ELMBs (Embedded Local Monitor Boards) [10]; one to readout temperature sensors and the other to supervise vacuum pressure. The high voltage is provided by a single channel ISEG power supply and distributed via a Hamamatsu distributor, providing 160 negative high voltage channels. Both ISEG and ELMB devices were already supported by the JCOP framework.

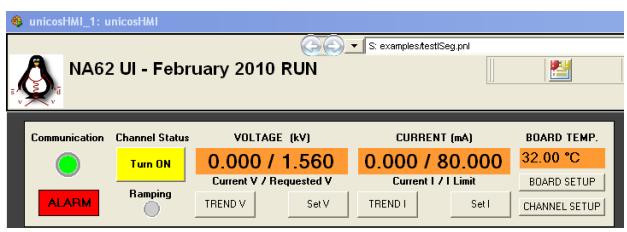


Figure 5: Extract from the LAV prototype DCS user interface.

The monitoring and control of the ELMBs and ISEG power supply was performed with WinCC OA and the combined framework approach mentioned in earlier sections. The high voltage distributor was controlled directly from a front-end program using a CAMAC controller and a CAMAC analog-digital converter.

In addition to the device level controls, a series of application level features were configured in the LAV

prototype DCS. These included an alarm screen, SMS notification of problems and checks on the DCS integrity.

LAV Environment Monitor

An additional application was required to provide long-term temperature monitoring of the area in which LAV stations are stored prior to installation. This application uses one ELMB with 32 PT100 sensors. The DCS application built upon the previous UNICOS-JCOP integration work for the LAV prototype DCS. Further work on this integration provided additional cohesion enabling the use of features such as the UNICOS trending tool. Other application features include access control and Oracle archiving to CERN central servers.

This application has been running in production for 6 months so far and was able to successfully detect of failure of the climate control system in the storage area.

CURRENT STATE AND FUTURE PLANS

In September 2011 the DCS for NA62 reached its first milestone: the completion of the "LKr (Liquid Krypton Calorimeter) Vertical Slice" application. This application provides integrated hierarchical control for custom high and low voltage systems of the LKr.

In this application, the synergy between the UNICOS and JCOP frameworks was raised to a higher level. The hardware that has typically been used in UNICOS, namely PLCs, has been brought under the supervision of the finite state machine based hierarchical control tool of the JCOP Framework and combined with analog value readout provided by JCOP ELMB devices. In addition, the hierarchical control tree has been embedded in the standard UNICOS user interface to act as the main navigation mechanism [11], providing the standardized look and feel compatible with other EN/ICE applications. The details of these integration efforts will be presented in a future paper.

The roadmap for future development of the NA62 DCS assumes that further synergy between the frameworks is achieved, so that a coherent and standardized engineering process could be established for future applications of a similar type. Based on the current experience and achievements of the "LKr Vertical Slice" milestone, a gradual evolution of the control application will proceed by including subsequent sub-detectors. An integration run in spring 2012 is planned to demonstrate the complete application in one quarter of its complete scale.

The work performed so far has demonstrated a significant interoperability between the frameworks. However, in order to reach the goal of closer integration, some topics will require further work:

- Further integration of the two device models.
- Widening of system integrity checks across the UNICOS and JCOP domains.
- Unified application generation including JCOP device instantiation.

CONCLUSIONS

The NA62 LAV and LKr applications have validated the approach to gain the benefits of both the UNICOS and JCOP frameworks. The device proxy concept enables the high level UNICOS tools to access JCOP data required at operations time. This provides a seamless interface to the control application operator.

Additional synergy can be obtained by integrating the frameworks together more closely. There are tools that are still limited to functioning in one of the two framework domains. As these issues are tackled, the full benefits of this combined framework approach will become available.

The further work required on the NA62 DCS will provide a suitable test bed and showcase for future integration work. There are still significant conceptual and design efforts ahead, but experience to date shows that the results of such work should be reusable in many future applications, providing a sustained improvement in efficiency of application building and maintenance.

REFERENCES

[1] F. Hahn (ed.) "NA62 Technical Design Document" http://na62.web.cern.ch/na62/Documents/TD_Full_doc_v10.pdf (2010).

[2] CERN Bulletin, "NA62 Gigatracker sets new standards for silicon detectors", Issue: 26-27/2011, Article ID: BUL-NA-2011-156 (2011).

[3] P. Golonka et al., "NA62 Requirements for the Control and Monitoring of the Experiment Subdetectors", http://edms.cern.ch/file/1038362/1.5/NA62_URD.pdf (2010).

[4] ETM professional control GmbH – WinCC Open Architecture, http://www.etm.at/

[5] J. Andersson et al., "The Controls Middleware (CMW) at CERN Status and Usage", ICALEPCS'2003, Gyeongju, South Korea (2003).

[6] C. Gaspar et al., "DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication", CHEP'2000, Padova, Italy (2000).

[7] W. Salter "LHC Data Interchange Protocol (DIP) Definition" (2004) http://edms.cern.ch/file/457113/2/DIPDescription.doc

[8] H. Milcent et al., "UNICOS: An open framework", ICALEPCS'2009, Kobe, Japan (2009).

[9] O. Holme et al., "The JCOP Framework", ICALEPCS'2005, Geneva, Switzerland (2005).

[10] H. Boterenbrood and B. Hallgren "The Development of Embedded Local Monitor Board (ELMB)", 9th Workshop on Electronics for LHC Experiments, Amsterdam, The Netherlands (2003).

[11] F. Varela and L. Petrova "Monitoring of Controls Applications at CERN", ICALEPCS'2011, Grenoble, France (2011).

DATABASE DRIVEN CONTROL SYSTEM CONFIGURATION FOR THE PSI PROTON ACCELERATOR FACILITIES

H. Lutz, D. Anicic, Paul Scherrer Institute, 5232 Villigen PSI, Switzerland

Abstract

At PSI there are two facilities with proton cyclotron accelerators. The machine control system for PROSCAN which is used for medical patient therapy, is running with EPICS. The High Intensity Proton Accelerator (HIPA) is mostly running with the in-house developed control system ACS. The control of dedicated parts of HIPA runs already under EPICS. Both these facilities are configured through an Oracle database application suite. This paper presents the concepts and tools which are used to configure the control system directly from the database-stored configurations. Such an approach has advantages which contribute for better control system reliability, overview and consistency.

INTRODUCTION

Control system configuration for HIPA and PROSCAN facility is kept in its own databases HIPADB and PRODB (see Fig. 1). The database information can be split into the following categories :

- Device Reference contains all information about the machine layout in terms of Device names, sections, positions, and their topology.
- IOC (Input Ouput Controller) configuration for EPICS. The IOC configuration for the ACS control system has been already described in detail in [1].
- Interlock configuration data which are used for the Run Permit system in both facilities HIPA and PROSCAN.
- Beamline Reference data are used to produce input files for the automatic beam control and for the simulation tool Transport as well.
- Archiver configuration for the in-house developed data archiver software.

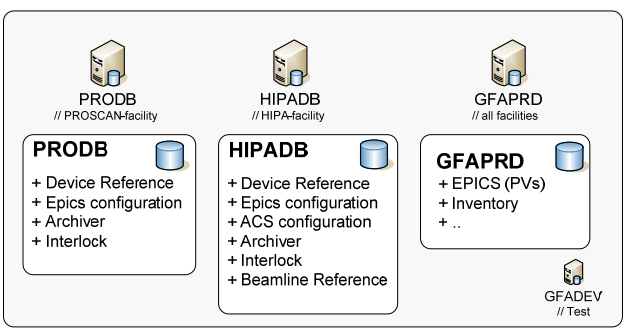


Figure 1: Oracle databases for HIPA/PROSCAN control system configuration.

A third Database GFAPRD displayed on Fig. 1 contains data used for all facilities in common. As an example these kind of information includes

- Hardware Inventory and Computer configuration.
- EPICS PV (Process Variable) names for all facilities.

DATABASE ENTRY

Figure 2 shows the building blocks for the database driven configuration. Oracle Webforms delivered from an Oracle Application Server are used in order to enter the data by the enduser.

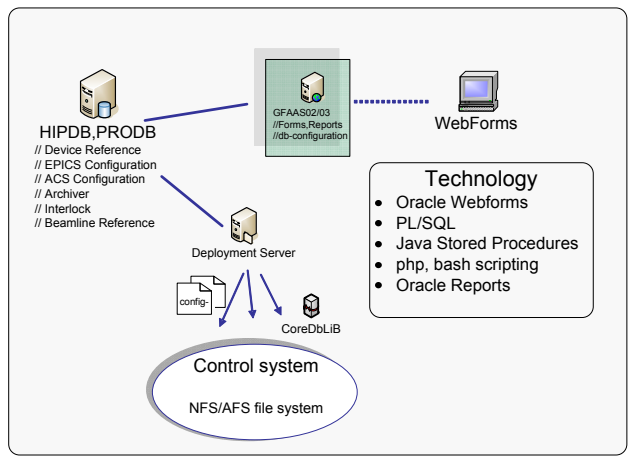


Figure 2: HIPA/PROSCAN Configuration Infrastructure.

PL/SQL libraries are used to provide procedures and functions for database extraction and Java Stored Procedures and php scripts are used to deploy the configuration files on to the NFS (Network File System) shared file system. The deployment itself, is done on a dedicated deployment server and is triggered directly from the Webforms.

The described method has been used in production for many years (since 2002). Several redesigns and adaption has been made in order to fulfil new requirements. One example is the EPICS configuration system which is used for the PROSCAN control system. There are different categories of information which are stored in several database schemas. Each category has its own Webform editor.

Device Reference Editor

Device Reference holds everything about definition and organization of all the elements in the PSI proton accelerator complex as seen by the operator. This may be a bending magnet, quadrupole, profile monitor, a pump etc. Each element has a unique name (up to 8 characters)

according to our Device Naming convention and is arranged into machine parts. Each element is connected to a parent according to the physical arrangement along the beamline. Device listings ("Holy Lists") may be produced from this information. The Device Reference tables are directly connected to the EPICS configuration and Beamline Reference tables.

EPICS Configuration, IOC Setup

The definition of the IOC configuration consists of defining a name, attributes like the EPICS version and a set of startup script lines. According to the placement (slot, channel) of hardware modules the corresponding DriverConfig strings must be filled in. For each module one or more configuration strings can be defined in the form DriverConfig (p1,p2,..). Each record in the DriverConfig table has a unique key which is used in the Template record definition. The configuration of Soft IOCs without connection to a piece of hardware is supported as well.

EPICS Configuration, Templates

With a Template editor we can setup and maintain a pool of EPICS templates with its records and field values and their logic (forward links etc). New templates can be uploaded from the NFS file system. Minor changes in the template structure (add a new record, field) can be handled by the editor itself. For each template the macro variables with default values can be extracted.

EPICS Configuration, Macro Substition

The substitution of the macro variables is the final stage of configuration. For a given Device eg. a magnet, the Device name is mapped to the corresponding IOC configuration slot together with the corresponding Template. From the Template Designer's point of view we have implemented two constraints. It's not allowed to have more then one references to the IOC configuration slot. Each Template must have at least one macro called $DEVICE. This macro is automatically replaced by the Device name which is optional connected to the Device Reference table. All other macro values can be manually defined in the substitution editor.

DEPLOYMENT & TEST

The deployment of IOC specific configuration files which can be triggered by the "install button" from the Webform is shown in Fig. 3. For the selected IOCs the following tasks are done on the deployment server :

- From the set of IOC configuration tables the common startup script together with a driver config script is generated.
- A unique substitution file for each IOC is generated which contains all the macro values with a reference to the corresponding template.
- At the end all files are deployed to the NFS file system structure and the IOC can be rebooted through the corresponding BootServer.

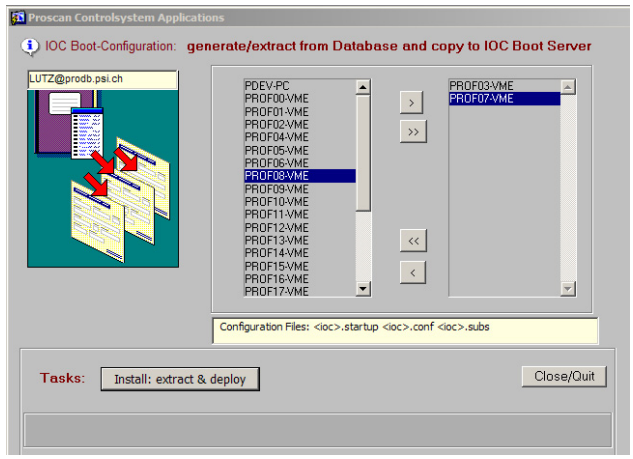


Figure 3: Deployment of IOC configuration files.

The database relevant tasks which are done on the IOC boot server are explained in Fig. 4. During the IOC boot process all the created record names together with some useful attributes like RecordType, IOC name, timestamp etc) are loaded into the EPICS PV database. New records are inserted, and records which do not exist anymore will be marked as deleted.

The information in the EPICS PV database is used by several database tools. Findrecords is a comandline tool which allows to query the PV database with several filter option. "duprecords" lists all duplicated records inside a facllity. "capv-view" is a Java application which lets filter all the PVs by IOC and Device name. After setting the filter all PVs values are displayed. The corresponding Web-Tool has been developed using Microsoft .NET technology. In the web-based Inventory tool the PV names together with the current values can be displayed as well. Database Reports are used to display or check the Device Reference-PV database relationship.

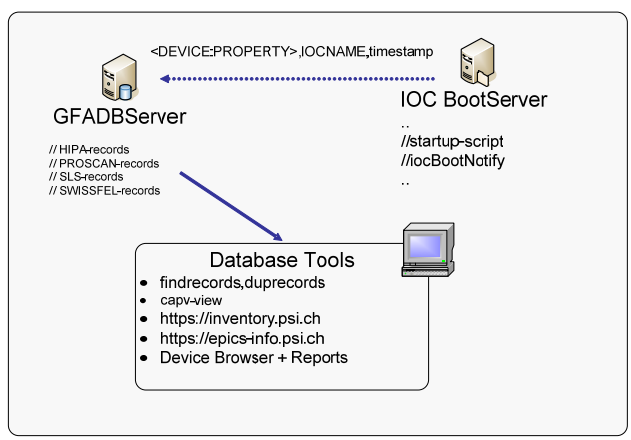


Figure 4: Loading EPICS record names during IOC boot process.

The following table shows some statistics about the number of EPICS IOCs, Devices and PVs in the PROSCAN control system. PROSCAN operates with 27 VME IOCs which are configured with 70 templates. Only 3 Soft IOCs are used. In summary PROSCAN is controlled with 1700 Devices and 50000 records. The values for the HIPA EPICS control system (eg. 50 IOCs + 100 templates) mentioned in table [1] are estimated values as most of the systems are running with the ACS control system.

Table 1: Statistics PVs IOCs Templates

DB (Facility)	PVs (record:device)	IOCs (vme+soft)	Templates
PRODB	50000:1700	27+3	70
HIPADB	100000:2700	50+5	100
HIPA values are estimated, 90% is still under ACS Control			

APPLICATION CONFIGURATION

Interlock Data

Interlock holds data tables for description of the Run Permit system. Interlock modules are organized into areas, groups and sections. Each module is mapped to an IOC slot and has configurable input/output function tables. The Interlock tables feed data input for the Interlock monitor application.

Beamline Reference Data

In order to specify input information for the calculation of beam envelopes using the Transport code [2] the Beamline Reference Editor is used to fill the data sets which are connected to the Device Reference tables. Each entry consists of a predefined Transport type code, driftspace and an optional value string. together with a 4 char Transport Label. Depending on the type code the required driftspace is automatically filled in. This configuration is mainly used in the HIPA facility. The input files can be generated and deployed to the file system on request. These files are used in the control room during Beam Setup and Tuning. A second customer is the automatic beam control application.

Archiver Configuration Data

For the HIPA/PROSCAN facility an in-house developed archiver software for data archiving is used. The configuration file with all the PV-Names together with the corresponding logging period (5 sec, 1 min, per hour, per shift) is configured by a dedicated Configuration Editor. The Editor is connected to the EPICS PV database and supports functions like adding/removing PVs and restart the archiver server which is running on a dedicated machine.

DATABASE TOOLS

Device Browser

With the Oracle Reports tool an EPICS Device browser based on html pages has been developed. On a start page the user can enter some query conditions and gets afterwards information about Devices, EPICS templates, macro substitions and IOC configurations.

Database Reports

Several Paper Layout reports have been built using Oracle Reports Builder. After building a report the corresponding executable (rdf-file) is delivered to the Application Server, see Fig 2. The Intranet Webuser can retrieve the new report with information in different output formats (html/pdf/xls). This includes Device Reference lists, dedicated information about Interlock database, Magnet parameter settings, database statistics etc. These Reports are permanently modified and built on request.

CONCLUSION

The database driven configuration has been successfully implemented in both facilities PROSCAN and HIPA for operation with two different control systems. There are many advantages for using this method :

- A lot of information about the configuration of the control system is kept in a single database instance.
- It helps a lot when upgrading tasks have to be done.
- The whole system is maintained by only a few persons (in our case 3-5).
- Applications are configured centrally. Additional applications can be easily integrated.
- Documentation of the system can be easily derived from database reporting.

Besides these positive aspects one negative aspect has to be mentioned. It is not easily editable as with text files on a file system. Therefore it is not well suited in a Test & Development scenario.

REFERENCES

[1] H. Lutz, D.Anicic et al "Migration of the Configuration Database for PSI Cyclotron Accelerators", ICALEPS'03, Gyeongju Korea, Nov 2003

[2] http://people.web.psi.ch/rohrer_u/trans

PRELIMINARY DESIGN AND INTEGRATION OF EPICS OPERATION INTERFACE FOR THE TAIWAN PHOTON SOURCE

Y. S. Cheng, Jenny Chen, P. C. Chiu, C. H. Kuo, C. Y. Liao, K. T. Hsu, C. Y. Wu
NSRRC, Hsinchu 30076, Taiwan

Abstract

The TPS (Taiwan Photon Source) is the latest generation 3 GeV synchrotron light source which has been in construction since 2010. The EPICS framework is adopted as control system infrastructure for the TPS. The EPICS IOCs (Input Output Controller) and various database records have been gradually implemented to control and monitor available subsystems of the TPS at this moment. The subsystem includes timing, power supply, motion controller, miscellaneous Ethernet-compliant devices etc. Through EPICS PVs (Process Variables) channel access, remote access I/O data via Ethernet interface can be observed by the useable graphical toolkits, such as the EDM (Extensible Display Manager) and MATLAB. The operation interface mainly includes the function of setting, reading, save, restore and etc. Integration of operation interfaces will depend upon properties of each subsystem. In addition, the centralized management method is utilized to serve every client from file servers in order to maintain consistent versions of related EPICS files. The efforts will be summarized at this report.

INTRODUCTION

The TPS [1] is a latest generation of high brightness synchrotron light source which has been under construction at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan since 2010. It consists of a 150 MeV electron Linac, a 3 GeV booster synchrotron, and a 3 GeV storage ring.

The EPICS (Experimental Physics and Industrial Control System) is a set of open source software tools, libraries and applications developed collaboratively and used to create distributed soft real-time control systems for scientific instruments such as the particle accelerators, telescopes and other large scientific experiments [2]. In the field of accelerators, many facilities have good practical experiences for EPICS and adopt it as the accelerator control systems. Many resources and supports are available as well as numerous applications for accelerator have been developed.

As a result, the EPICS framework was also selected as control system infrastructure for the TPS project. The EPICS platform has been gradually built and tested to control and monitor the subsystems of TPS. The various database records can be created for accessing the I/O data and setting parameters at the IOC (Input Output Controller) layer. Adopting the EPICS channel access mechanism with specific toolkits, the data can be accessed between the IOCs and the clients.

During the implementation process of the EPICS support for various subsystems, the operation interfaces of each subsystem are also developed according to the different operation methods. To simulate the operation process, the various operation interfaces are needed to integrate. The centralized management of the EPICS related files is also adopted, and the mechanism of save and restore will be continuously developed. The efforts will be summarized as following.

SOFTWARE ENVIRONMENT

The client consoles are adopted the Linux operation system. All of the EPICS base, modules and extensions are installed at the Linux system. The software versions are shown as Table 1 where most of applications are developed base upon this software environment.

Table 1: Software Environment of the Control Consoles

	Version
OS	RHEL 5.4 (32-bit) (kernel 2.6.18)
EPICS	base-3.14.10
Modules	asyn-4.11a StreamDevice-2.4
Extension	edm-1.12.xx labCA-3.1

FILES MANAGEMENT

All EPICS related files at control consoles are mounted from the file server by using the NFS service [3] to simplify software version control. Various directories are created and saved into various versions of related files for various hosts and purposes. Various directories provide a mount point for hosts mounted according to various purposes. The directories include EPICS base, modules, extensions, saved data, temporary data and etc.

Several file servers are established to share the loading of NFS file service. The hosts mount specific file server according to its location and purpose. By loading testing, the NFS file service is divided into three parts. Two servers provide the NFS service for hosts of all cells; the other server is for engineer development.

To keep the same version at each NFS file server, the "rsync" function is used with the "crontab" schedule service [4]. One main NFS file server is established for development purpose; the other NFS file servers are adopted "rsync" function to synchronize all data from the main NFS file server with using the private Ethernet

communication. The system architecture is shown as the Fig. 1. The dedicated spare NFS file server is also established for redundant purpose, and the related data is synchronized from the main file for backup purpose.

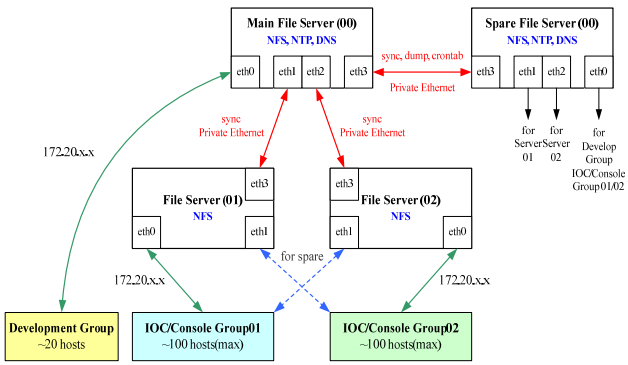


Figure 1: The system architecture of file servers for EPICS related files mounted with backup mechanism.

SAVE AND RESTORE

To readily restore a set of the machine parameters for subsystems during operation as well as to optimize and record working points for different machine condition, the mechanism of save and restore is developed. The save and restore function is initially built by using the MATLAB with labCA [5]. The various files of grouped PVs (Process Variables) list are created for saving the respective parameter values of each subsystem. The file with PVs and saved parameters is also selectable for resume the settings. The preliminary interface of save and restore mechanism is shown as the Fig. 2. Both of consumed computer resource and process time are acceptable after preliminary testing.

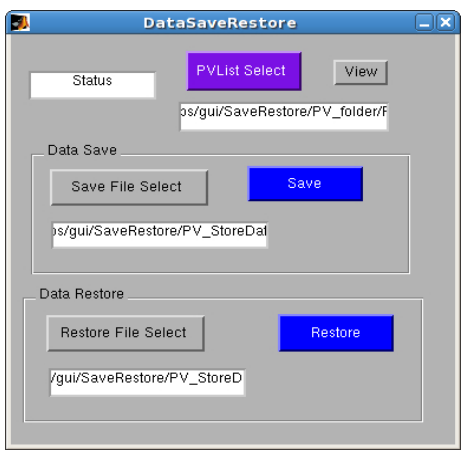


Figure 2: The operation interface of save and restore.

PRELIMINARY CONTROL PAGE

At the development phase, the GUI of TPS control system adopts the EDM (Extensible Display Manager) toolkit to develop main graphical operation interface. The EDM pages of various subsystems are created and saved into the NFS file sever for client console operation.

The EDM is an interactive GUI builder and execution engine, the EPICS documentation uses the term display manager, and maintained by the ORNL EPICS community [6]. All "objects" of EDM are loaded from shared libraries. The EDM administrator can add and remove objects from the list of available objects without recompiling EDM itself. The objects are versioned, carefully coded objects can be upgraded without impacting existing displays.

The preliminary main control page is built by the EDM toolkit shown as the Fig. 3. All control pages can be launched from this GUI. All control components are located at the foreground of the TPS accelerator illustration. For example, the LTB (Linac to Booster) dedicated control page is linked from the main control page for operation shown as the Fig. 4. The related control parameters or components are also located at the same page for tuning easily, such as the control page for the TPS timing as shown in the Fig. 5. These pages will be continuously refined and developed for the future commissioning.

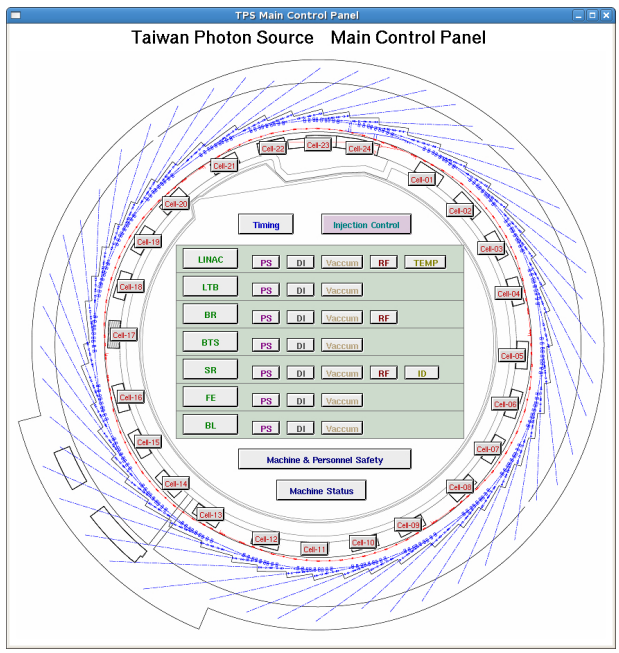


Figure 3: The preliminary TPS main control page.

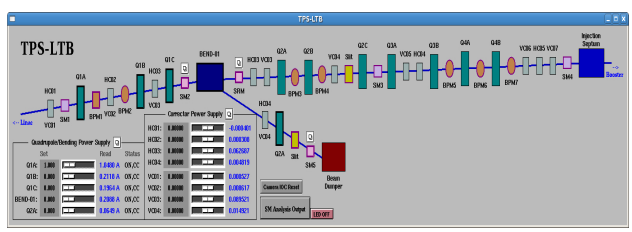


Figure 4: The LTB control page for operation test.

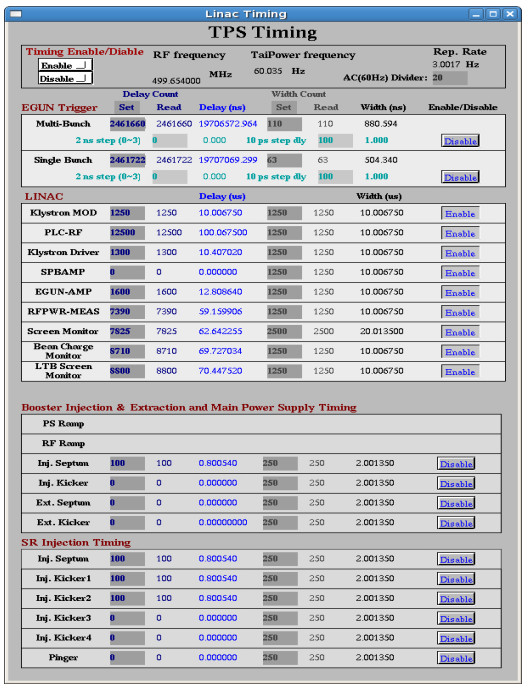


Figure 5: The overview control page for the TPS timing.

POWER SUPPLY OPERATION INTERFACE

The EDM toolkit is also adopted to develop the operation interface of power supplies. The client console can operate the specific EDM page to access the data via PVs channel access. The preliminary GUI page of storage ring dipole, quadrupole and sextupole power supplies controls as shown in Fig. 6. The Fig. 7 shows the GUI page of storage ring vertical corrector, horizontal corrector and skew quadrupole power supplies controls. The macro name method was regularly used to switch each display page. The main control page was shown critical information for observing status easily, and the main operation process functions are also executed from the panel.

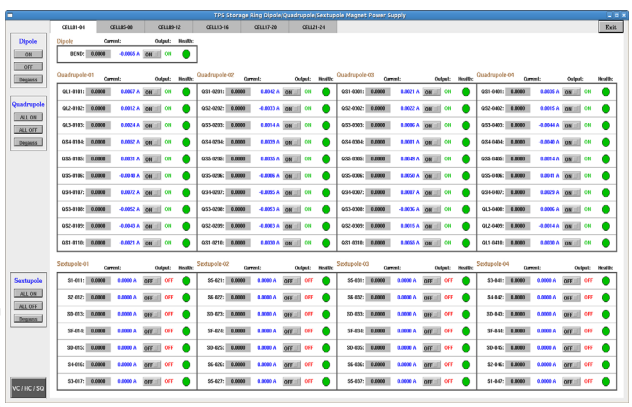


Figure 6: The control GUI of storage ring dipole, quadrupole and sextupole power supplies.

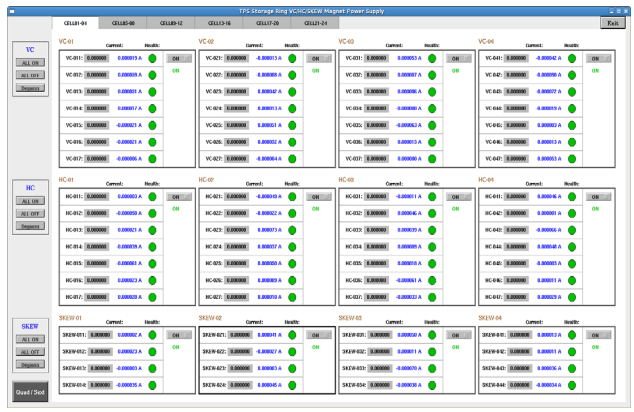


Figure 7: The control GUI of storage ring vertical corrector, horizontal corrector and skew quadrupole power supplies.

The MATLAB toolkit with labCA is adopted to develop the high level application programs for commissioning and diverse operational procedures. The applications include the specific overall power on/off control, degauss process, checking power supply status, operation performance analysis, operation statistics and etc. The various operation processes will be developed and tested according to the various operation modes. The detail control page of power supplies has the trend plot for observing. The Fig. 8 shows the current variation during the degauss process executed. The degauss application is also developed with the specific function of batch process to reduce the peak of power consumption for saving energy.

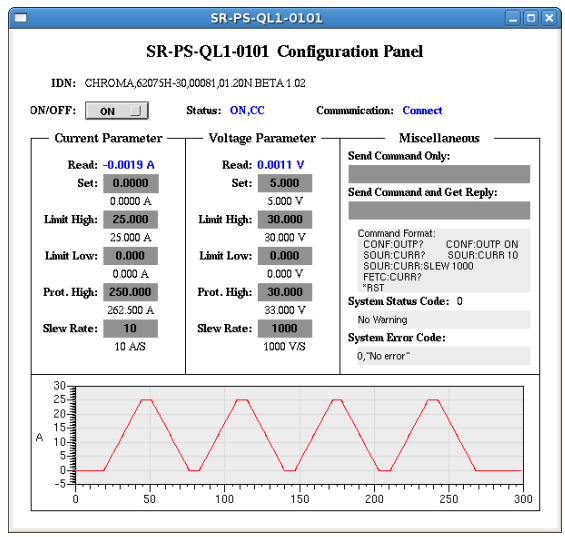


Figure 8: The current trend is shown at configure page of a quadrupole power supply during the degauss process test.

ARCHIVE

To record the variation of specific subsystem for long time observation, the archive can be use to save data. The EPICS channel archiver [7] is under research and evaluation. The archive view is the java based toolkit, and

can be easy to use at the different operating system platforms. The server loading and saved files size of archive system will be estimated in the future.

SUMMARY

The TPS control system adopts the EPICS framework as the control infrastructure. The EPICS supports of various subsystems are built gradually. The operation interfaces of various subsystems are also developed in the meantime. The various GUIs are designed according to the different operation methods. The management of the EPICS related files system is also established for maintaining files versions easily. The mechanism of save and restore are needed to apply parameters for operation process. Various operation interfaces will be integrated and improved continuously during the TPS control system construction.

REFERENCES

[1] http://www.nsrrc.org.tw/english/tps.aspx.
[2] http://www.aps.anl.gov/epics/.
[3] http://en.wikipedia.org/wiki/Network_File_System_(protocol).
[4] http://kevin.vanzonneveld.net/techblog/article/synchronize_files_with_rsync/.
[5] http://www.slac.stanford.edu/~strauman/labca/.
[6] http://ics-web.sns.ornl.gov/edm/.
[7] http://ics-web.sns.ornl.gov/kasemir/archiver/

NEW SPring-8 CONTROL ROOM: TOWARDS UNIFIED OPERATION WITH SACLA AND SPring-8 II ERA

A. Yamashita*, R. Fujihara, N. Hosoda, Y. Ishizawa, H. Kimura, T. Masuda, C. Saji,
T. Sugimoto, S. Suzuki, M. Takao, R. Tanaka, JASRI/SPring-8, Hyogo, Japan.
T. Fukui, Y. Otake, RIKEN/SPring-8, Hyogo, Japan

Abstract

We have renovated the SPring-8 control room. This is its first major renovation since its inauguration in 1997. In 2011, the construction of the SACLA (SPring-8 Angstrom Compact LAser) was completed. Plans are to control it from the new control room for it to work in close cooperation with the SPring-8 storage ring. It was expected that the upcoming SPring-8 II project would require more workstations than the current control room could accommodate. We have therefore extended the control room area for these anticipated requirements. In this renovation, we employed new technologies that did not exist 14 years ago, such as a large LCD and silent, liquid-cooling workstations for a comfortable operation environment. We have incorporated many ideas which were obtained during the 14 years experience of the operation. Operations in the new control room were started in April 2011 after a short period of construction.

INTRODUCTION

The SACLA (SPring-8 Angstrom Compact free electron LAser) [1], which has been constructed in the SPring-8 campus, successfully generated 0.12 nm wavelength X-ray laser in June 2011, three months after the start of its commissioning run. The SACLA has an unique feature that allows experimentalists to use synchrotron radiation from SPring-8 and an X-ray laser from the SACLA simultaneously and perform an experiment such as a pump-and-probe experiment. And also, the SACLA is designed to act a high performance injector for SPring-8. Therefore, it is planned that SPring-8 and the SACLA will be controlled from the same SPring-8 control room for a closely coupled and unified operation. They are controlled from separate control rooms during the commissioning period. We are also planning major upgrade for SPring-8 which is named SPring-8 II [2] to be implemented over the next decade. We expect SPring-8 II will require more workstations to control. The previous control room did not satisfy this requirement. For this purpose, we decided to renovate the SPring-8 control room. Fig. 1 shows a view of the new control room.

When the old control room was constructed in 1997, there were no liquid crystal displays (LCD), water-cooled, silent workstations, or large LCD panels for display wall, which are popular for use in modern control rooms. We renovated the control room to take advantage of the those modern technologies that did not exist in 1997.

* aki@spring8.or.jp

DESIGN GUIDELINES

Before the design, we set the design guidelines as follows.

- SPring-8 and SACLA should be controllable from the same control room.
- The control room should be scalable for implementing SPring-8 II.
- The control room should be such as that there is no partition between the two accelerators.
- The control room should have function in cooperation with the safety and facility systems.
- The control room should have consoles and the numbers of machine control CPUs (e.g. VME) should be as low as possible for quiet environment.

We designed the new control room according to the design guidelines.

FLOOR LAYOUT

Fig. 2 shows the floor layout of the new control room. We divided the control room area and the back area using the display wall. Approximately half of the old control room was the back area and was filled with racks for servers and workstations. Extenders were used to connect workstations in the back area with keyboards, mouses and displays in the control area. This arrangement helped reduce the noise generated by workstation and keep the control area quiet for maintaining a good working environment. We had to place each workstation at the back of the control desk in the new control room because the extender and the long and thick cables resulted in a maintenance problem. The use of a silent type of workstation solved the noise-related problem. We reduced the size of the back area and moved the servers to the newly constructed dedicated server room.

Twenty-four control workstations could be distributed among five desk *islands*. One island in front of the display wall had four main terminals for an overall SPring-8 accelerator control and one safety interlock panel with the emergency stop button, access keys, and a display panel for displaying interlock status. Terminals from the other islands were needed to monitor and control individual accelerator components. The fish-shaped desk islands design enabled the operators at the main terminals to view information related to all the terminals at a glance and have face-to-face communication with to all the other operators. In addition, the fish-shaped design is scalable as required by the design guidelines and it can be extended by adding extra desks.

The number of consoles was selected based on current usage for SPring-8 and the SACLA plus the expected re-

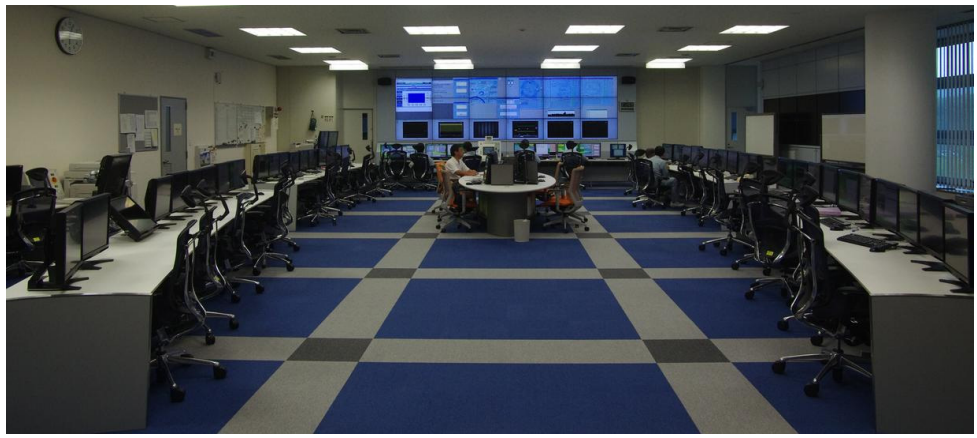


Figure 1: View of the new SPring-8 control room.

quirement in the future. Currently, SPring-8 accelerators and beam lines are controlled by less than 17 workstations in the control room. There are more than 7 spare spaces for workstations.

In the center oval table accommodated six PCs. They were used for office work. We accommodated six thin client terminals on the side desk island. These terminals were used for monitoring the data acquisition status and the alarm. They were also used for program development.

Control Desk

Each control desk measured 1150 mm (Width)×855 mm (depth) for one operator. The depth was reduced for LCD screen because the old desk had a depth to fit a large CRT display. The width was sufficient to place for setting two 24" diagonal LCD displays side by side. The desks were built such that they were jointed with each other. The desk had space for a workstation at the back. The back panel could be removed without using any tools for workstation maintenance (Fig. 3). The workstation was water cooled and of the silent type for reduced noise. There were sufficient space at the back for a typical middle tower type case. In addition to the two LCDs, the desk was designed to fit four panels. Two holes on the side supported the poles for LCD stand. Fig. 4 shows the four panel setup.

Meeting Space

There was no dedicated meeting space in the old control room. We created meeting space with 12 chairs. Sometimes access to the accelerator was required during meetings. For this reason, the meeting space and the control section were not separated by a partition. The table had AC power sockets for note PCs belonging to individual participants.

Center Table

The oval table in the center had six PCs, which were used for office work such as for writing reports, reading documentations, and a simple analysis. We added a shelf under the oval table to place recent logbooks and manuals. The shelf helped to minimize the number of papers scattered on the desks.

Figure 3: Back view of the control desk. A workstation is built in. Two lines of AC power are supplied. One is stabilized by UPS for the workstation and the other is not stabilized.

Monitoring Space

The alarm and data acquisition systems were monitored from the side island. The thin client terminal displayed the processes performed at servers. Thin client terminals occupy very little spaces but their graphic display performance is poor. Tasks that do not require very good display performance are performed with the help of thin client terminals.

Back Area

Currently, the back area has very few components unlike the old control room, which was filled with server and

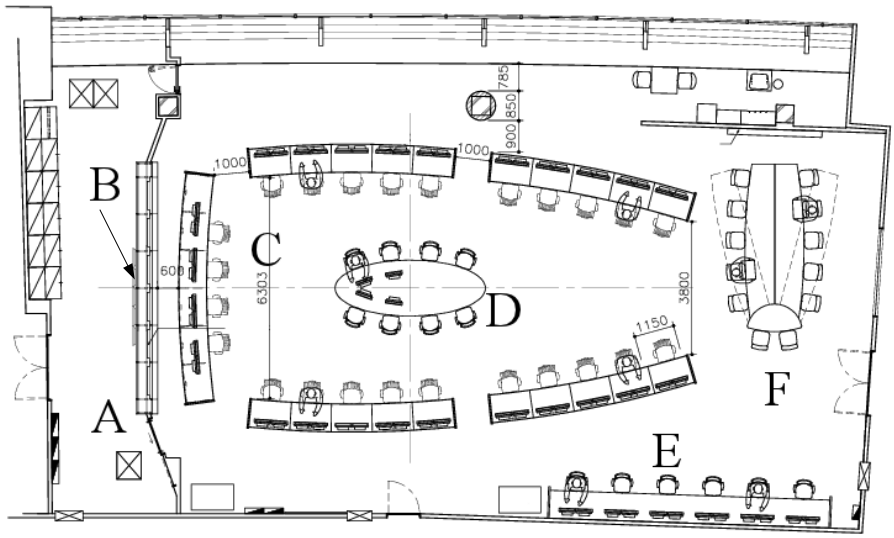


Figure 2: Floor plan of the new SPring-8 control room. A: Back area. B: Display wall, C: Control desks, D: Oval desk, E: Monitoring desks and F: Meeting space.

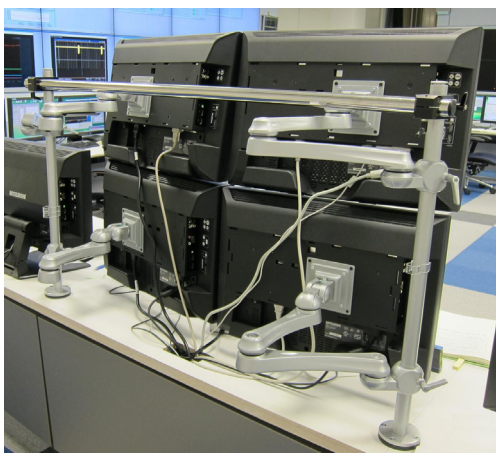


Figure 4: A desk has two poles to support 4 LCD panels.

workstation racks. The current back area is almost empty space aside from the three racks for networking and interlocks and sliding shelves for document archives. The operator sometimes resets the interlock signal on the rack, but this is relatively rare. Placing interlock equipment in the back area is convenient for the operator and satisfies the design guideline, which require the number of machines to be as low as possible.

Cabling

The control room was constructed on a 50 cm height raised floor to enable easy cabling. We removed almost all the cables that has been accumulated in the old control table like stratum, set cable ladder under the raised floor and routed cables.

TECHNOLOGIES

Built-in Workstations

In the old control room, workstations were placed in the back area and connected to the keyboard, video monitors and the mouse via an extender and thick cables to reduce the noise in the control section. Workstations were placed at the back in the control desk to allow easy maintenance. The use of the water-cooled, silent but high-performance workstation eliminated the noise-related problem owing to its noise level of less than 30 dB. The reliability of the water-cooled workstation is quite acceptable, given that none of the twenty workstation have experienced any water leakage problem even after half a year operation. The specifications of the workstations are shown in Table 1 Those workstations are suitable for SPring-8 control tasks.

Table 1: Workstation Specifications

Item	Value
CPU	Intel®Xeon®X3450 2.66 GHz Quad core
Memory	4 GB
Disks	250 GB × 2 Raid1

Networks

SPring-8 networks are classified as office LAN, DMZ LAN, control LAN and SACLA control LAN. Firewalls between them control the data flow. In the control room, we limited the use of the wired control LAN, wired SACLA LAN and the wireless office LAN. None of workstations are connected to more than one LAN simultaneously. PCs used for office work are connected to the office LAN using wireless connection even when they are desktop PC. Dividing the network into wired and wireless simplifies network cabling and reduces the chance of incorrect wiring. We concealed network ports inside the desk to avoid the problems due to an unexpected PC connection.

New Display Wall

In the previous ICALEPCS 2009, we presented a display wall hiring a 6x2 configuration[3]. We have designed another 6x3 configuration display wall to display more information that will be required in SPring-8 and the SACLA combination operation. The old display wall has been shifted to the side of the room. The new display wall employs an ultra-narrow bezel (7 mm between the screen and the next screen) 46" diagonal LCD[1] . The previous display wall was driven by a cluster consisting of seven PCs. On the other hand, the 18 panels of the new display wall are driven by just one PC equipped with three graphics cards. Each graphics card has 6 mini-DisplayPort sockets[2] , Currently MS-Windows is installed on the driver PC, because its display driver software has better stability and usability than that of the Linux OS. As shown in Fig. 1, the display wall displays accelerator status, alarm status, video images and more to share information between operators.

Analog TV Signal Viewer

Previously we monitored the screens of oscilloscopes, spectrum analyzers, and beam monitors which installed close to the accelerators after converting their information to analog NTSC video signals. In the renovation, thee control room was designed to use only digital data and analog signals were removed from the control room. Camera servers[3] that converts NTSC signals to motion JPEG are attached to the source of the video signals and video images are transmitted using the HTTP protocol. Although standard web browsers can display such images, we developed a dedicated viewer using web APIs provided by vendors [4] and used widget libraries [5] for easy operation and to save the display area and the ability of motion JPEG files to record data.

[1] NEC LCD-X462UN
[2] ATI Firepro™ V9800
[3] AXIS 24x series

CONSTRUCTION

The construction for the renovation was carried out during the spring shutdown period of 2011 so as to not affect the SPring-8 operation. We reduced the duration of construction by completing as many tasks as possible before the construction.

Cable ladders were installed under the raised floor during the summer shutdown period of 2010. In addition, equipment for interlocks was moved during the same period. We also examine cables which were unrecorded their end points before the construction.

Approximately 40 days were required for removing old furniture and cables, replacing the old display walls, painting walls, refurbishing the floor, constructing new display walls, installing new furniture, changing the lighting and etc. The construction was complete on schedule, and we have been operating SPring-8 from the new control room since April 2011.

SUMMARY

The new SPring-8 control room has been constructed within a short period. We conformed to the design guidelines as follows. The fish-shaped desk islands met the scalability requirement for the SACLA and SPring-8 II, The interlock panel placed at the center of the consoles and the interlock signal at the back area enabled close coordination between the safety systems and the accelerators operations, and the division between the control area and the back area helped to create an environment specific to the control area.

ACKNOWLEDGMENT

The authors would like to thank Mr. T. Hamano for his effort in constructing the display wall.

REFERENCES

[1] H. Tanaka, "Status Report on the commissioning of the Japanese XFEL at SPring-8," Proceedings of IPAC 2011, San Sebastion, Spain, 2011.

[2] T. Watanabe et al., "Current Status of SPring-8 Upgrade Plan," Proceedings of IPAC 2011, San Sebastion, Spain, 2011.

[3] T. Hamano et al., "Development of large high-resolution display for SPring-8 central control room," P.111, Proceedings of ICALEPCS2009, Kobe, Japan, 2009.

[4] Axis VAPIX Version3 manual, `http://www.axis.com/files/manuals/VAPIX_3_HTTP_API_3_00.pdf`.

[5] wxPython, `http://www.wxpython.org/`.

THE LHC SEQUENCER

Reyes Alemany-Fernandez, Vito Baggiolini, Roman Gorbonosov, Denis Khasbulatov, Mike Lamont, Pascal Le Roux, Chris Roderick, CERN, Geneva, Switzerland

Abstract

The Large Hadron Collider (LHC) at CERN is a highly complex system made of many different sub-systems whose operation implies the execution of many tasks with stringent constraints on the order and duration of the execution. To be able to operate such a system in the most efficient and reliable way, the operators in the CERN control room use a high level control system: the LHC Sequencer. The LHC Sequencer system is composed of several components, including an Oracle database where operational sequences are configured, a core server that orchestrates the execution of the sequences, and two graphical user interfaces: one for sequence edition, and another for sequence execution. This paper describes the architecture of the LHC Sequencer system, and how the sequences are prepared and used for LHC operation.

THE LHC SEQUENCER ARCHITECTURE

The LHC Sequencer Architecture is made of two core components: the Sequencer Executor and the Database. Two Graphical User Interfaces (GUI), one to interface the executor and another one to interface the database, provide the operators in the CERN Control Centre (CCC) with the required control on running sequences and on the creation or modification of sequences, respectively.

The Sequencer Executor part has been extensively presented in [1]. Here a small summary of the main components will be recalled, but emphasis will be given to the database part and the operational sequences used in the daily life operation.

Sequencer Server and Sequencer Client

The architecture and technology used by the Sequencer follows the CERN accelerator controls software standards: a 3-tier architecture implemented in Java using the Spring Framework [2]. This architecture is shown in Figure 1. The sequencer middle-tier server (running Linux) contains the core functionality, mainly, sequence execution, temporary sequence storage and interface via Java libraries to the controls of the different LHC sub-systems, i.e. the LHC Software Architecture (LSA) [3] and the accelerator controls Common Middleware (CMW) [4].

The client tier consists of an execution GUI, shown in Figure 1, from which the execution of the sequences is controlled by the LHC operators in the Linux or Windows consoles at the CCC. Many GUIs may connect to the same middle-tier server.

Database Persistent Storage of Sequences

All sequences are persistently stored in the Oracle database. The database schema consists of a series of tables that map the sequences representation in the following way:

- Sequence table: stores the sequence name, the display name as will appear in the GUIs, brief description, date of creation, name of the person that created the sequence and the sequence category. Each sequence is uniquely identified by a primary key (PK); the name is constraint to be unique since the sequencer server retrieves sequences by name, not by PK.

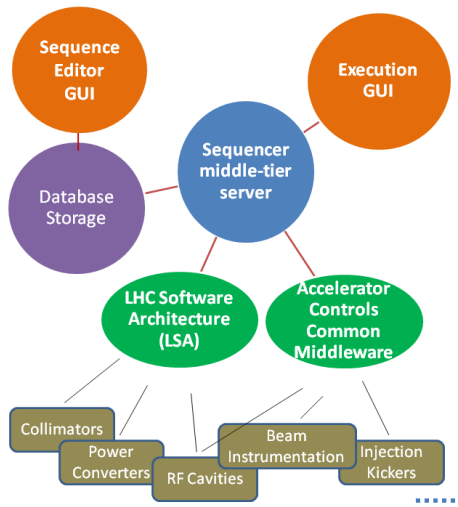


Figure 1: LHC Sequencer Architecture.

- Subsequence table: stores the same information as the sequence table. Each subsequence is uniquely identified by a primary key, and, for the same reason as above, the name is unique.
- Seq-subsequence table: each sequence is made of one or several subsequences. This table contains the list of subsequences that belongs to each sequence. A subsequence can be used by different sequences
- Sequence_components table: each subsequence is made of a number of components which can be of two types: atomic tasks or subsequences. Each atomic task component can be described by a number of parameters which corresponds to columns in the same table. Those parameters are: the name of the hardware group to be addressed by the task (for example a name referring to all the power converters in LHC); tasks can act on a group of hardware or on a single device, therefore another parameter is the device name; the name of the LHC cycle during which the task will have to act, e.g. LHC.USER.INJECTION, LHC.USER. RAMP. If the component is a task, there are a set of separate tables that describe those tasks, uniquely identified by a

primary key which is referred to within the Sequence_components table. Each task component can be further configured to have certain behaviour in case the task fails the execution. The behaviour can be the execution of another task, or another subsequence, or a complete sequence, or simply, stop or continue. Four columns in the components table are reserved to define the on_error behaviour. Once the on_error behaviour is executed by the server, the main sequence can continue execution or stop. This is configured in the same components table via another column. Finally, components are assigned an execution order, an integer number that the server uses to establish the execution order of all the tasks inside the (sub)sequence. If two or more components of type task have the same order, they are executed in parallel at the server level, and only when all the parallel tasks are finished (either successfully or with errors) the server returns the control to the GUI for that particular (sub)sequence. The tasks or subsequences have a default action that can be run, skip or break. All components configured to run are executed (following the established order) when the (sub)sequence is executed; if a component is configured to be skipped, then it will not be executed; if a component is configured as a break component, the execution of the (sub)sequence will be stopped at this component.

- Category table: each (sub)sequence is categorized according to the following criteria:
    - Physics: final version of a (sub)sequence, which has been debugged and it is ready to be used in routine operation.
    - Development: (sub)sequences under construction and debugging, not to be used during routine operation until they are validated and become operational, i.e. of physics category.
    - Machine development: (sub)sequences dedicated to machine development periods.
    - Equipment specific: (sub)sequences to test specific accelerator equipment for commissioning.
- Task type table: within this table, general task configuration parameters can be specified, i.e. parameters that precise if the task will be executed using LSA methods or it is a CMW type task.
- Task instance and task instance parameters tables: once a task type is defined, instances of it can be further created. Each instance can be parameterized to act on a particular property of the task type. For example, a task type frequently used in LHC sets the beam mode. The task type is called SET BEAM MODE and it is a LSA task. Several instances of it exist in the database, each sets the mode to a given value, e.g. SET BEAM MODE TO INJECTION PROBE BEAM, SET BEAM MODE TO RAMP, etc. Within the task instance parameters table the different modes are specified which are columns of the table called, parameter_name and parameter value. Parameter_name in this example is called mode, and the parameter_value is the mode name: INJECTION PROBE BEAM, RAMP, etc.

One of the most important advantages of using the database as persistent storage platform is that the sequences can be edited, modified and new sequences, subsequences and task instances can be created without the need of releasing any software component. The database provides with a very dynamic user interface.

Sequence Editor and Sequence Executor

Every of these tables and columns is filled using the Sequences Editor, a Java program that access directly the database via SQL statements. The editor is protected with Role Based Access [5] in order to restrict the edition of sequences to authorized people only. A picture of the Sequence Editor is shown in Figure 2. On the left hand side panel the user can select the sequence to be modified. There is a filtering panel in order to show only the sequences of a given category. From this panel access to task type and task instances tables are possible, as well as the possibility to create a new sequence. On the right hand side panel different Java tables map the tables in the database and different buttons allow adding the required information. The Java tables are editable.

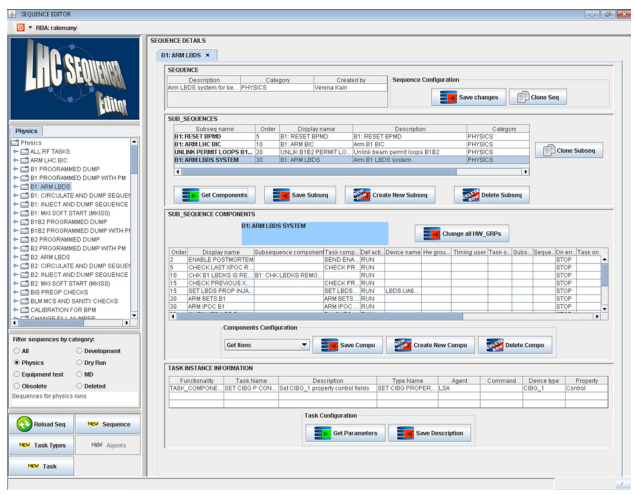


Figure 2: LHC Sequence Editor GUI.

Several of all those columns presented before are freign keys to columns within the LSA database tables. For example, the hardware group column of the table sequence_components, is filled with the content of the hardware_group column of a particular LSA table containing this information. In this way the user cannot enter any name which could not be recognized by the system, but the Editor shows, in the form of a combobox, only the allowed hardware_groups.

Once a (sub)sequence is created or modified by the user, the sequencer server retrieves it from the database and it is converted into a Java source file. The right compilation of the source code is a way of ensuring, to a great extent, the coherence of the sequence.

The Sequencer Executor is a graphical user interface (GUI) developed using the Standard Widget Toolkit[6]. Figure 3 shows a picture of the Executor.

The operator can search for a given sequence, or using the Quick Launch Panel (Figure 4) can search for the most used sequences which are organized by functionality, e.g. Specific Equipment sequences, Experiment sequences, etc. Once a (sub)sequence is selected, the GUI shows the corresponding tree structure of the sequence where the component names displayed are the ones defined by the user as displayed names in the database tables. Clicking on a particular task, the parameters of the tasks, as read from the database, are shown in the tab called "Details". The GUI allows for drag and drop of subsequences.

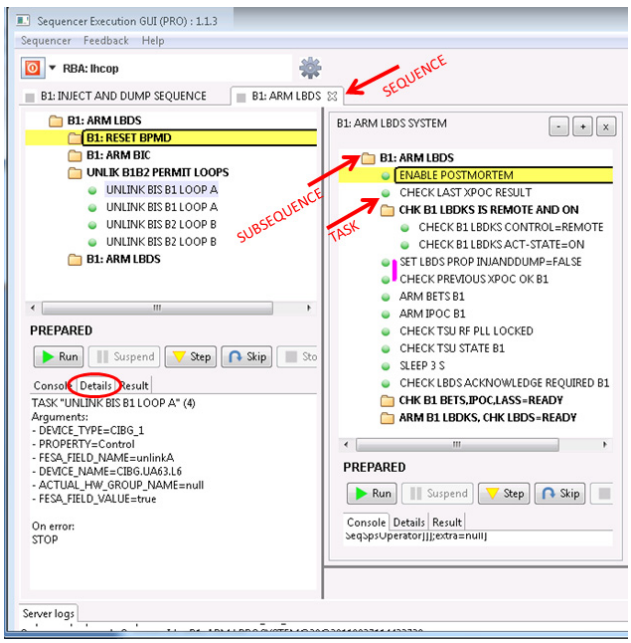


Figure 3: LHC Sequence Executor GUI.

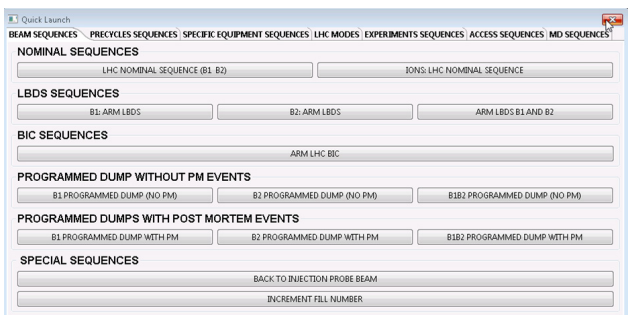


Figure 4: Quick Launch Panel.

In Figure 4 the subsequence B1: ARM LBDS has been dragged and dropped on the right and can be executed as an independent sequence. The vertical magenta bar in the picture indicates that the two tasks called SET LBDS PROP INJANDDUMP=FALSE and CHECK PREVIOUS XPOC OK B1 will be executed in parallel since they have the same execution order in the database. The default action configured in the database (run, skip or break) can be modified online within the executor GUI, but it is not propagated to the database. Modifications of the database configuration can only be done with the Editor. The subsequences can be (un)collapsed to show more or less details of the tree.

USE OF SEQUENCES FOR LHC OPERATION

The LHC operation in routine mode, like luminosity production, certain machine development periods, machine optimization studies, relies fully in the execution of sequences; nothing is done "manually". In the following we explain two cases rather representative of the LHC operation: Inject and dump sequence and LHC nominal sequence.

Inject and Dump Sequence

The distinctiveness of this sequence is that uses a particular functionality of the executor server, the possibility of coming back to a given task after completion of the execution and re-starting again the execution in an automatic way. This functionality is achieved via two tasks, one labels the position within the sequence from where the execution will have to be repeated, the other task instructs the executor to go to the label task. This functionality is very important in this sequence because it allows the arming of the beam dump system and the beam interlock system, injection into LHC and dump in a continuous way when performing injection steering or injection studies.

LHC Nominal Sequence

The LHC nominal sequence contains more than 1000 tasks and is organized in a set of subsequences that maps the LHC cycle: preparation for injection, injection, ramp, squeeze, collisions, and after more than 10 hours of stable beams, programmed dump and ramp down, as depicted in Figure 5. The LHC nominal sequence addresses all the LHC equipment at different steps in the cycle. It is not run in one go as the case above, but every subsequence is run separately, or step by step, since in between subsequences, beam measurements and corrections have to be performed which are not included in the sequence.

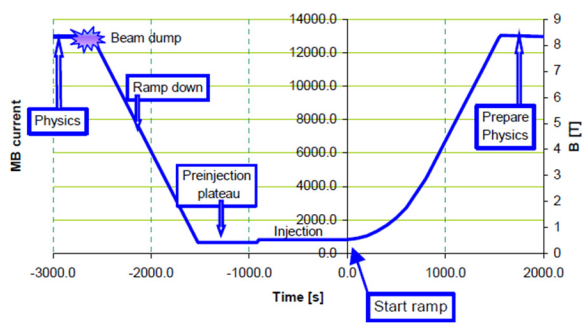


Figure 5: LHC nominal cycle (for 7 TeV flat top). All the different steps in the cycle are driven by the LHC Sequencer.

REFERENCES

[1] V. Baggiolini, R. Alemany Fernandez, R. Gorbonosov, D. Khasbulatov, M. Lamont, "A Sequencer for the LHC era", ICALEPCS'09, Kobe, Japan, 2009, Conference Proceedings.
[2] http://www.springframework.org
[3] G. Kruk, S. Deghaye, M. Lamont, M. Misiowiec, W. Sliwinski, "LHC Software Architecture [LSA] – evolution toward LHC beam commissioning", ICALEPCS'07, Knoxville, Tennessee, 2007, Conference Proceedings.
[4] http://proj-cmw.web.cern.ch/proj-cmw/documents.htm
[5] S. Gysin, A.D. Petrov, P. Charrue, W. Gajewski, V. Kain, K. Kostro, G. Kruk, S. Page, M. Peryt, "Role-based Access Control for the Accelerator Control", ICALEPCS'07, Knoxville, Tennessee, 2007, Conference Proceedings.
[6] http://www.eclipse.org/swt

AUTOMATED VOLTAGE CONTROL IN LHCb

L. Granado Cardoso, C. Gaspar, R. Jacobsson, CERN, Geneva, Switzerland

Abstract

LHCb is one of the 4 LHC experiments. In order to ensure the safety of the detector and to maximize efficiency, LHCb needs to coordinate its own operations, in particular the voltage configuration of the different sub-detectors, according to the accelerator status.

A control software has been developed for this purpose, based on the Finite State Machine toolkit and the SCADA system used for control throughout LHCb (and the other LHC experiments). This software permits to efficiently drive both the Low Voltage (LV) and High Voltage (HV) systems of the 10 different sub-detectors that constitute LHCb, setting each sub-system to the required voltage (easily configurable at run-time) based on the accelerator state.

The control software is also responsible for monitoring the state of the Sub-detector voltages and adding it to the event data in the form of status-bits. Safe and yet flexible operation of the LHCb detector has been obtained and automatic actions, triggered by the state changes of the accelerator, have been implemented.

This paper will detail the implementation of the voltage control software, its flexible run-time configuration and its usage in the LHCb experiment.

INTRODUCTION

As an LHC experiment [1], LHCb relies on the LHC machine to provide the conditions for the production of events. In order to acquire the data from the events that happen in the experiment, LHCb has an infrastructure composed of several subsystems, namely several sub-detectors to record different parameters of the events. This infrastructure is sensitive to the type of configuration of the LHC accelerator and can be damaged if it's not configured properly according to the LHC status. As the experiment is built underground and is composed of many custom built detector components, it is not easily accessible nor can the parts be easily replaced in case of damage.

Furthermore, the efficiency of the detector must be optimized in order not to lose important beam time and record the most number of events with the best possible quality. In order to have the best efficiency, the LHCb detector must be configured as soon as possible into a state that matches the requirements for each particular LHC status.

As the LHCb detector is composed of several sub-detectors, for a given LHC status each of them must be configured in a particular state, which often differs between them. The configuration of all of them into the required settings one by one would prove inefficient as well as prone to errors.

A control software was developed that can, for each of the LHC states, drive each of the sub-detectors equipment into the appropriate voltage level and can easily be configured according to the sub-detector needs.

As this control software knows the current status of the voltages it is also able to calculate it in the form of status bits, which is then added to the event data and provides more information regarding the quality of the recorded events.

VOLTAGE CONTROL

The voltage control of the LHC experiments is based on the PVSS SCADA system [2], which interfaces the drivers of the equipment to the Finite State Machine toolkit [3].

The LHCb sub-detector voltage control is modelled into a FSM tree which has as the bottom nodes the modelled hardware devices (HV/LV) which are then grouped as sub-detector HV and LV systems, grouped then into logical units which contain all the sub-detectors HV/LV Voltage systems; these are all under the top node ECS (Experiment Control System) [4].

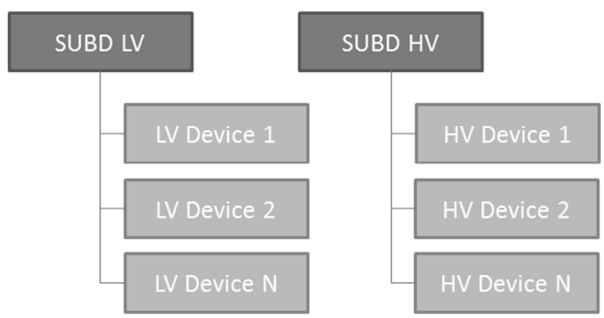


Figure 1: Typical sub-detectors voltage control FSM trees.

The commands flow down the tree, i.e. an action sent to the sub-detector top node flows down to the devices which perform it, while states flow up the tree, i.e. a state change at the device level is propagated up to the top nodes.

For the automated voltage control actions are sent to the top level nodes which group the HV and LV systems of each sub-detector.

Voltage FSM Control Units

The voltages in the LHCb experiment are controlled via the FSM control hierarchy and in order to set a particular voltage system into the desired state, commands are sent to the respective system top control node.

The nodes that group all the voltages for each of the systems (HV/LV) and each sub-detector are FSM Control Units and they can be in the following discrete states:

For the Low Voltage:

- NOT_READY – when not all of the low voltage devices are ON
- READY – when all the low voltage devices are ON
- OFF – when all the low voltage devices are OFF
- ERROR – when any of the low voltage devices is in ERROR
- EMERGENCY_OFF – when any of the low voltage devices has switched OFF due to an emergency action or trigger

For the High Voltage, as the time it takes to reach the appropriate voltage level is not negligible, the states can be of 2 types: stable states and transition states.

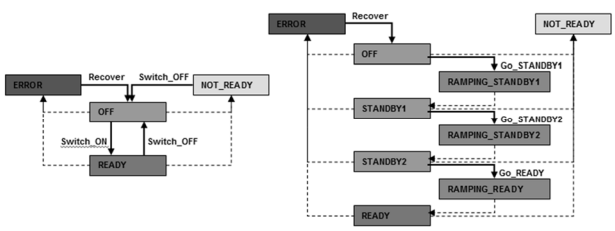
For the stable states, the voltage systems can be in the following states:

- NOT_READY – when not all of the high voltage devices are in a stable state
- OFF – when all the high voltage devices are OFF
- STANDBY1 – when all the high voltage devices are STANDBY1
- STANDBY2 – when all the high voltage devices are STANDBY2
- READY – when all the high voltage devices are READY
- ERROR – when any of the high voltage devices is in ERROR
- EMERGENCY_OFF – when any of the high voltage devices has switched OFF due to an emergency action or trigger

The transition states are called RAMPING_<Stable_State> and are set whenever the voltages are moving to that particular stable state.

The actions available to these Control Units are of the type GOTO_<Stable_State> and can take a parameter RUN_TYPE which allows for the selection of the proper recipe for the configuration settings of the devices.

These FSM objects serve as templates for the sub-detectors to model their voltage systems and allow for a transparent integration of its systems into the global LHCb Control System.



(a) DCS Domain (b) HV Domain

Figure 2: Simplified Voltage Control FSM Domains.

LHC State

The LHC machine publishes the status of the accelerator to all the experiments at CERN via DIP (Data Interchange Protocol).

LHCb subscribes to the LHC data and computes the state of the LHC accelerator, using this data; a FSM Device Unit models this data into 12 discrete states, 10 for different LHC configurations and 2 for error reporting ("ERROR", "UNKNOWN). This Device Unit can then be included in the FSM control hierarchy and be used as a trigger for automated actions as well as information for other FSM Units in the hierarchy.

Some of the computed LHC states are:

- PHYSICS – stable beams are declared and LHC is running with its nominal parameters
- PHYS_ADJUST –
- INJECTION – the beams are being injected into the LHC accelerator, the detectors voltages need to be in a safe state.
- MD – Machine Development; the LHC is adjusting operation parameters in order to optimize operation (e.g. cleaning, injecting probes). The detectors voltages need to be in the safest state possible
- DUMP – the beams have been dumped from the LHC accelerator

The actions available on the automated voltage control system can then be configured according to each of these individual computed LHC states instead of particular voltage states.

For the most dangerous states of the LHC, in respect to possible damages to the front end systems, LHC has implemented a handshake system to insure that the operation in these states is acknowledged by the experiments and they have set their systems to safe parameters of operation.

AUTOMATED VOLTAGE CONTROL

The automated voltage control system developed takes advantage of the FSM control hierarchy and is based on the FSM toolkit.

By using the voltage control FSM objects templates implemented by the sub-detectors, which are integrated into a global control FSM, and knowing the possible LHC states, it is possible to develop a control software which can configure the voltage systems in coordination with the specific needs for each LHC state.

For each sub-detector voltage system, you can define, for each of the available LHC states, in which voltage setting the respective voltage system should be set. The voltage settings can be any of the available states for that voltage system as well as a state called "_ANY_". In this case the sub-detector voltage system is allowed to be in any of its available voltage settings (safe for ERROR) and the state of this system in respect to the LHC state will always be OK. This is to allow for independent control from the part of the sub-detectors, when the LHC is in a state that allows for periods of development (e.g.

"NO_BEAM") without interfering with the global state of the experiment voltage systems control.

The configuration of the voltage states required for each sub-detector and each LHC state is easily done via a configuration table, which is also useful as an overview of the settings.

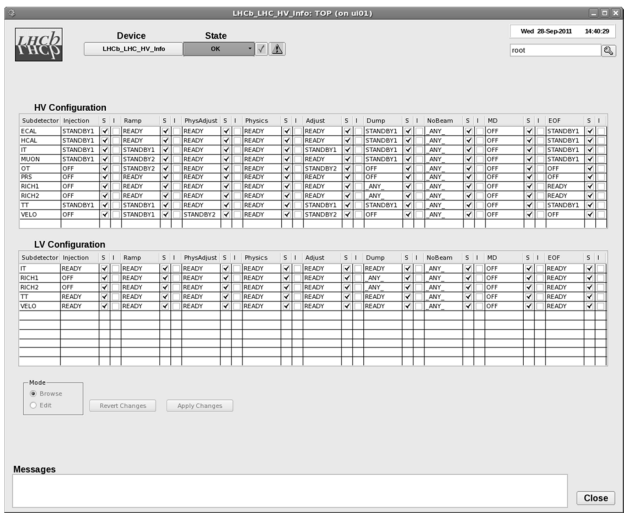


Figure 3: Configuration table (LHC state – LHCb voltage settings).

The Automated Voltage Control software is based on a set of FSM objects which will now drive the voltage systems, acting as a proxy between requests for configuration for a particular LHC state and the required HV configuration and a new FSM tree built upon these units and with references to the particular voltage systems to control.

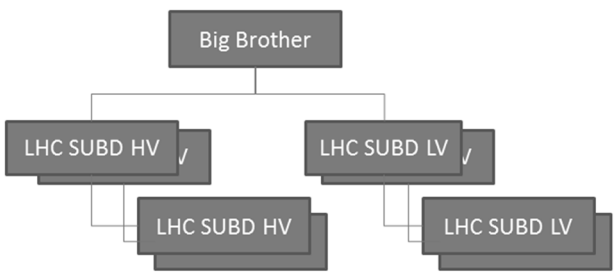


Figure 4: LHCb – LHC voltage control FSM Tree.

This new FSM tree will be the main control system for the LHCb sub-detectors voltage systems. It should have a top node with the CUs (Control Units) for the Voltage Systems to control as children. Typically each of these CUs is a sub-detector voltage system and references, as its children, the sub-detectors FSM voltage objects. These references serve as configuration for the new FSM object which will now drive the voltages.

The new discrete states of all these FSM objects are now a reflection of the state of the sub-detectors voltage systems in respect to the possible LHC states defined for the LHC state device unit. The actions are now, as well, commands to configure the voltage systems according to the configuration needed for each particular LHC state. This means that that the actions are no longer e.g. "GOTO_STANDBY" but e.g. "GOTO_INJECTION", and the voltages of each sub-detector will be configured according to the setting which corresponds to the "INJECTION" LHC state. Also the transitional states represent the movement of the voltage devices into the final requested LHC state configuration settings (e.g. "RAMPING_INJECTION").

Device Units

There are 2 Device Unit types needed to properly run the voltage systems.

One retrieves the parameter RUN_TYPE from a PVSS datapoint, in order for it to be correctly defined when the action for the voltage systems to go to a particular LHC state is requested.

The other Device Unit is the object that now drives the voltage systems. This Device Unit holds the proper configuration of all the required voltage settings for each LHC state per sub-detector and voltage system (e.g. "LHC_MUON_HV", "LHC_OT_LV").

There are also some other possibilities for configuration settings for each of the actions to be sent in respect to each particular LHC state request:

- Send action – The action can be set not to be sent until a user manually does this. The user does not need to know the correct voltage state needed but it does need to intervene in order for the voltage systems getting configured to the proper state.
- Ignore State – The action can be configured to automatically exclude the voltage system.

This Device Unit, on startup, connects to the states of the voltage system references of the particular sub-detector it controls and its state is computed from the accordance of the state of the voltage systems with respect to the last requested LHC state configuration. Its states are defined as follows:

- UNKNOWN
- ERROR
- RAMPING_<LHC_STATE>
- WAIT_MAN_CONFIG – Manual Configuration for this sub-detector for this LHC state is required
- OK – State of the Voltage systems IS in accordance with the needed state for this LHC state
- NOT_OK – State of the Voltage systems IS NOT in accordance with the needed state for this LHC state

High Voltage Status Bits

This Device Unit also computes a set of status bits for each High Voltage system, which is then saved with the events data in order to provide more information of the quality of the events.

For each sub-detector a majority limit threshold for the number of not working HV channels is defined and the status bits computed are based on the information of the individual channel states of each sub-detector and whether the number of incorrectly configured channels is greater than the majority threshold defined.

The status bits are as follows:

- 00 – All HV channels are correctly configured and running;
- 01 – There are some HV channels in an incorrect state in regards to the LHC state, but their number is below the majority threshold set.
- 10 – The number of channels in an incorrect state in regards to the correct state is above the defined majority threshold

The computation of the status bits is done by a thread which is launched for each controlled voltage system. This thread monitors periodically the state of each sub-detector High Voltage channel states and sets the status bits.

LHCb OPERATION

The Automated Voltage Control System implemented in LHCb is connected to a handshake system with the LHC. The LHC state based voltage operation allows this integration to be simplified as, whenever the LHC state changes and requests a handshake to move to a particular state, a simple action is the only required input from the part of the operator. A PVSS panel displays the readiness of all the sub-detectors Voltage Systems for the current LHC state. In this panel it is also visible the percentage of the each sub-detector which is in the correct voltage state (in terms of number of channels).

Further automation has also been implemented, and several actions can now be auto-piloted, providing a greater level of efficiency.

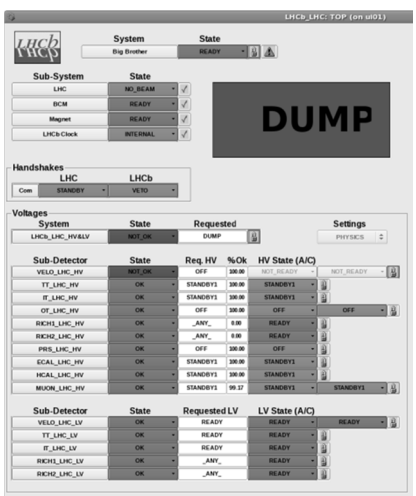


Figure 5: LHCb – LHC voltage control PVSS Panel.

CONCLUSION

The control software developed to drive the voltage systems in LHCb in accordance to the required settings for each of the LHC states proved to be a very efficient way of running the voltage systems of the experiment. The Automated Voltage Control is able to remove complexity from the system and improve the operation of the Experiment Control System as there is no need to track the different voltage settings for each of the sub-detectors at the different LHC states.

The safety of the detector is also assured as incorrect and unsafe voltage configurations can now easily be viewed in the main panel of the LHCb voltage control system and adjusted according to the current LHC state requirements.

The development approach based on the already implemented control system via the FSM toolkit also proved to be very reliable and configurable as well as scalable. Other voltage systems that use the LHCb FSM voltage control objects can easily be added.

This system also provides a base for further automation as LHC state changes can automatically trigger the correct voltage settings in a way very easily integrated in the current control system.

REFERENCES

[1] LHCb Collaboration, “LHCb Technical Proposal”, 1998.

[2] “PVSS II”, http://www.pvss.com/.

[3] C. Gaspar, “PVSS & SMI++ Tools for the Automation of large distributed control systems”, ICALEPCS 2005

[4] C. Gaspar and B. Franek and R. Jacobsson and B. Jost and N. Neufeld and et al, “An integrated experiment control system, architecture, and benefits: the LHCb approach”, IEEE Transactions on Nuclear Science, vol. 51, p. 513 2004.

SPIRAL2 CONTROL COMMAND: FIRST HIGH-LEVEL JAVA APPLICATIONS BASED ON THE OPEN-XAL LIBRARY

P. Gillette, E. Lemaître, G.Normand, L. Philippe, GANIL, Caen, France

Abstract

The Radioactive Ions Beam SPIRAL2 facility will be based on a supra-conducting driver providing deuterons or heavy ions beams at different energies and intensities. Using then the fragmentation and ISOLD method, exotic nuclei beams will be sent either to new physics facilities or to the existing GANIL experimental areas. To tune this large range of beams, high-level applications will be mainly developed in Java language. The choice of the OPEN-XAL application framework, developed at the Spallation Neutron Source (SNS), has proven to be very efficient and greatly helps us to design our first software pieces to tune the accelerator. The first part of this paper presents some new applications: "Minimisation" which aims at optimizing a section of the accelerator; a general purpose software named "Hook" for interacting with equipment of any kind; and an application called "Profils" to visualize and control the Spiral2 beam wire harps. As tuning operation has to deal with configuration and archiving issues, databases are an effective way to manage data. Therefore, two databases are being developed to address these problems for the SPIRAL2 command control: one is in charge of device configuration upstream the Epics databases while another one is in charge of accelerator configuration (lattice, optics and set of values). The last part of this paper aims at describing these databases and how java applications will interact with them.

THE SPIRAL2 FACILITY

Overview

The SPIRAL2 facility (Fig. 1) is based on a high power, superconducting linac driver, which will deliver a high intensity, 40 MeV deuteron beams as well as a variety of heavy-ion beams with mass-to-charge ratio of 3 and energy upon to 14.5 MeV/nucleon. A possibility of construction of a second injector for heavy-ions with a mass-to-charge ration of 6 is incorporated in the design. The main RIB production scheme of SPIRAL2 is based on the fast-neutron induced fission of uranium target. Using a carbon converter, a 5 mA deuteron beam and a high-density (up to 11g/cm^3) 2.3 kg uranium carbide target, the fission is expected to reach a rate up to 5*10^{13}/s. A direct irradiation of the UC_2 target with beams of protons or 3,4He could also be used. The extracted 1+radioactives ions will be injected in the 1+/n+ charge breeder (ECR ion source) and post-accelerated by the existing CIME cyclotron.

The SPIRAL2 project is divided in two phases (Fig 1): the first one (LINAC buildings and associated experimental AEL) and the second phase (RIB production building and DESIR facility).

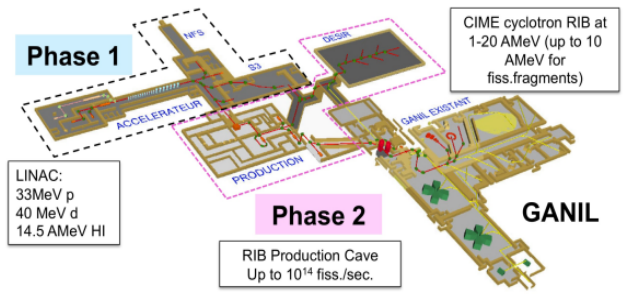


Figure 1: Accelerator layout.

SOFTWARE ARCHITECTURE

Control System Main Choices

The choice of EPICS [1] as a common framework was early decided to ease pieces of software development and integration efficiency. Mainly remote terminal units are VME VxWorks crates with MVME 5500 CPUs and Red Hat Enterprise Linux PCs hosting EPICS Input Output Controllers (IOCs). On the other side, Siemens S7 programmable logic controllers (PLC) are mainly dedicated to slow material protection systems needed for radio frequency, cryogenic, vacuum, or interlock systems.

Software Development

To ease software sharing, a unique EPICS SPIRAL2 working environment, an equipment naming convention, and operational rules for interfaces have been specified. To survey a large facility as SPIRAL2, supervision screens developed with EPICS Extensible Display Manager (EDM) are needed[2]. An archive engine using the one provided by Control System Studio (CSS) integrated development environment has been set up this year. For the tuning of the accelerator, more sophisticated high level applications are involved. XAL [3],[4],[5] Java based library and framework originally developed at the Spallation Neutron Source (Oak Ridge National Laboratory), and dedicated to accelerator tuning, have been adopted. The accelerator hierarchy and model design, fulfilling the Spiral2 specific needs, have proved to be very effective.

Among the numerous tools included in XAL, the high level API for wrapping EPICS Channel Access protocol are widely used to read, write and monitor values on devices.

Some news applications have been written or adapted from the ones already provided by the SNS.

THE MINIMIZATION APPLICATION

This software acts on a set of equipments to achieve the minimization of objectives on a set of diagnostics.

Some algorithms (Simplex, Powell, Coordinated steps, One dimension, Aleatory, Aleatory by map, Gradient) have been implemented, improved (when needed) and tested in order to face the situations that could occur during the tuning of the beam. It is based on standard XAL Framework and library tools, using node selector and PV selector, Scorer and AlgorithmSolver classes [6].

The application is divided in four parts: the user chose which active and passive elements he wants (up left), enters the start and the end desired with some options (up right), and then chooses and sets up the algorithm (bottom left). The results are drawn in real time in the same frame. The numerical results appear in the bottom right.

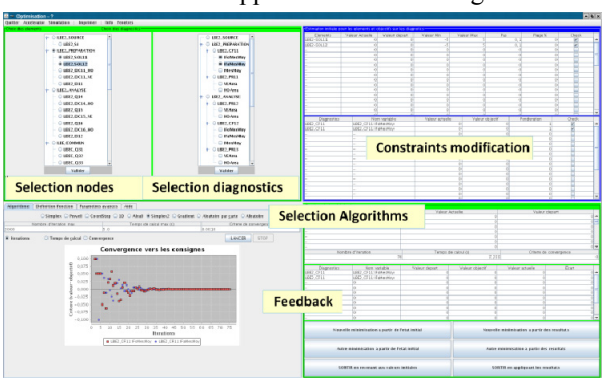


Figure 2: The minimisation software.

THE HOOK APPLICATION

A general purpose software named "Hook" has been developed for directly interacting with equipment of any kind (for the moment power and high voltage supplies, harp wire monitors, motorisations, current diagnostics, slits). For each kind of equipment, a generic component is used to assign elements on a graphic grid and provide complete access to the equipment. It was designed to reflect the same look and feel as the one (written in Ada) used in the legacy control system of GANIL.

Using the tools of the XAL library, the user selects elements in a hierarchical presentation of the accelerator. Once the elements are selected, each one can be assign to a sizable cell of a grid which the user can interact with. The grid component is in charge of the realisation and destruction of the items views.

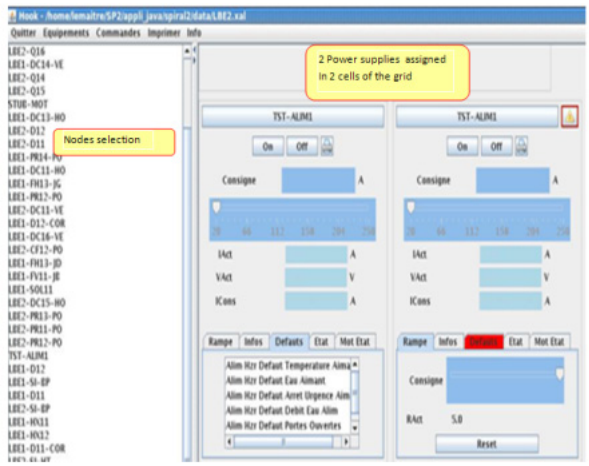


Figure 3: The hook software.

THE PROFILS APPLICATION

To visualize and control the SPIRAL2 beam wire harps the application Profils has been developed. It offers a complete control over the electronic devices and specific calculations on the data read from IOC Profils. It allows the user to select several sections of a profile to calculate areas, half widths, centers etc. The results of calculation are available to Epics clients via virtual process variables generated by a Java server.

As for the hook application, a set of harps can be assigned on a grid. General purposes commands are then available for all the harps assigned (scaling, printing, saving, insertion into the beam line, integration time or high voltage setting, mean and offset set-up etc)

Individual commands are also available in the window of each element: electronic device set-up, graphical comparison between live, theoretical, saved or snapshot data, selection of the wires to take in account for drawing and calculation, fit method for reconstruction of broken wires etc.

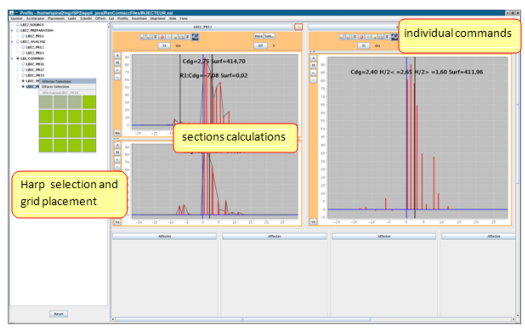


Figure 4: The Profils software.

DATABASES AND DATA MANAGEMENT

The command control group of GANIL is accustomed to databases usage on order to manage data and configuration linked to the system. The first Ingres database has been introduced in 1990.

For Spiral2 project, four topics rely on databases:

- Devices configuration
- Accelerator configuration
- Alarms handling
- Archiving

Device Configuration

To describe and exploit the 2500 equipments which are envisaged for the control of the SPIRAL2 facility, a relational database and associate management tools are developed at GANIL.

This database will be used offline and describes "simple" equipments such as slits, motors or power supplies. It enables generation of IOC configuration file (connections, substitutions and sequencers definition) as shown in Figure 5.

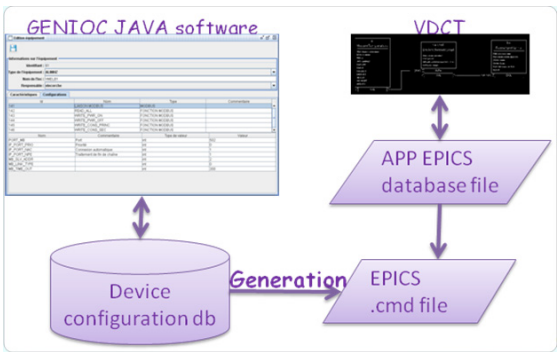


Figure 5: Device configuration process.

The main concept of this database is to describe a device type dynamically into the database's tables and instantiate this type to create a new device.

A first mock-up has been developed and tested with slits and power supplies devices.

Accelerator Configuration

To manage Accelerator configuration, a relational database and associate management tools are developed at GANIL

This database will be in charge of accelerator configuration. It describes accelerator hierarchy, optics and enables generation of XAL file defining the accelerator .xdxf.

The hierarchy is made up of 4 levels shown in Figure 7:

- Level: setting unit for operation. A level is made up of Paths.
- Path: path followed by the beam. A path belongs to a unique Level. A path is made up of sequences.
- Sequence: Grouping by tuning function. A sequence belongs to one or many paths. A sequence is made up of parts.
- Part: It's a unit introduced to facilitate computing management. A part belongs to one or many sequences. A part is made up of nodes (devices that affect or survey the beam).

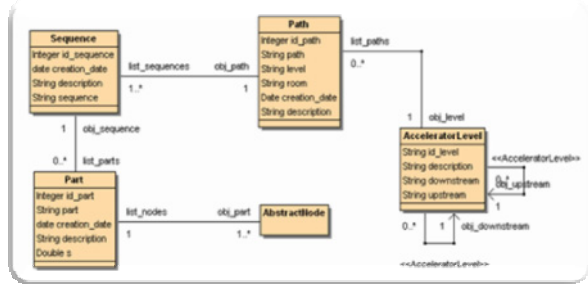


Figure 6: Accelerator hierarchy.

This database can be read and update off-line by a new dedicated software which enables generation of XAL file defining the accelerator optics files.

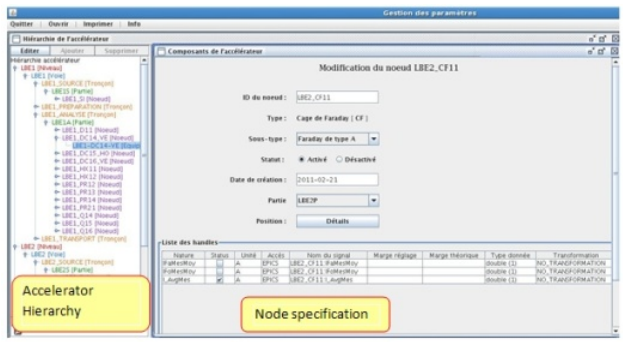
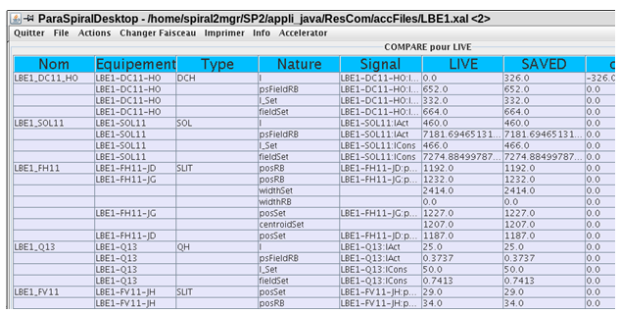


Figure 7: Accelerator configuration database tool.

This database will be used off-line to assign values for a new beam and to schedule operation. The set of values of a new beam is either theoretical or extrapolated from beams realized previously. Theoretical values for a beam to be produced are exported from Tracewin software developed by D.URIOT (CEA) [7].

This database will be used online to apply or save sets of values.

An application dedicated to the setting of the accelerator allows the user to choose the injector to use, the beam to accelerate and the set of values to apply. Once these settings have been applied, numerous operations (setting, reading, comparing values, computing data for new energies or magnetic rigidities) are available. 50-60 percent of functions are implemented


ParaSpiralDesktop - /home/spiral2mgr/SP2/appli_java/ResCom/accFiles/LBE1.xal <2>

Quitter File Actions Changer Faisceau Imprimer Info Accelerator

COMPARE pour LIVE

Nom	Equipement	Type	Nature	Signal	LIVE	SAVED
LBE1_DC11_HO	LBE1-DC11-HO	DCH	I	LBE1-DC11-HO:I...	0.0	326.0
	LBE1-DC11-HO		psFieldRB	LBE1-DC11-HO:I...	652.0	652.0
	LBE1-DC11-HO		I_Set	LBE1-DC11-HO:I...	332.0	332.0
	LBE1-DC11-HO		fieldSet	LBE1-DC11-HO:I...	664.0	664.0
LBE1_SOL11	LBE1-SOL11	SOL	I	LBE1-SOL11:IAct	460.0	460.0
	LBE1-SOL11		psFieldRB	LBE1-SOL11:IAct	7181.69465131...	7181.69465131...
	LBE1-SOL11		I_Set	LBE1-SOL11:ICons	466.0	466.0
	LBE1-SOL11		fieldSet	LBE1-SOL11:ICons	7274.88499787...	7274.88499787...
LBE1_FH11	LBE1-FH11-JD	SUT	posRB	LBE1-FH11-JD:p...	1192.0	1192.0
	LBE1-FH11-JG		posRB	LBE1-FH11-JG:p...	1232.0	1232.0
			widthSet		2414.0	2414.0
			widthRB		0.0	0.0
	LBE1-FH11-JG		posSet	LBE1-FH11-JG:p...	1227.0	1227.0
			centroidSet		1207.0	1207.0
	LBE1-FH11-JD		posSet	LBE1-FH11-JD:p...	1187.0	1187.0
LBE1_Q13	LBE1-Q13	QH	I	LBE1-Q13:IAct	25.0	25.0
	LBE1-Q13		psFieldRB	LBE1-Q13:IAct	0.3737	0.3737
	LBE1-Q13		I_Set	LBE1-Q13:ICons	50.0	50.0
	LBE1-Q13		fieldSet	LBE1-Q13:ICons	0.7413	0.7413
LBE1_FV11	LBE1-FV11-JH	SUT	posSet	LBE1-FV11-JH:p...	29.0	29.0
	LBE1-FV11-JH		posRB	LBE1-FV11-JH:p...	34.0	34.0

Figure 8: ParaSpiral draw a comparison between online setting values and setting values saved.

Alarms Handling

For reasons of compatibility and consistency between GANIL alarm handling process and that of SPIRAL2, we decide not to use EPICS standard Alarm Handler.

The alarms handling process, illustrated in Figure, rely on an Ingres database. This database assumes two main functions:

- Configuration of the process : how to build alarms messages and which alarms should be caught
- Storage of alarms issued.

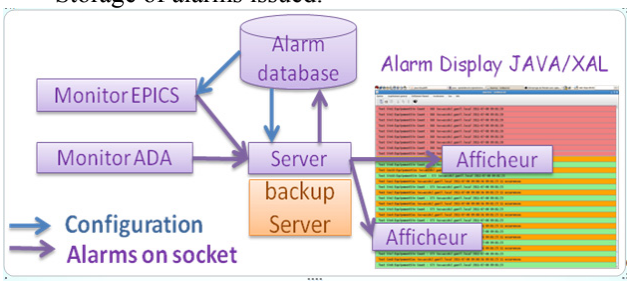


Figure 9: Alarm handling process.

The alarms process is made up of four software:

- Monitor EPICS: JAVA based development. It relies on channel access monitor. It is in charge of catching and formatting alarm message.
- Monitor ADA: ADA Based development. It assumes the same function as Monitor EPICS for GANIL alarms.
- Server: JAVA Based development. It receives alarms from monitor and dispatches them to displays. A failover server takes over if the main server failed.
- Display: JAVAL/XAL based development. It displays alarms in the form of button list.

This process has been tested and supports a throughput of 330 alarms per second.

Archiving

The archiving system is planned to be implemented witch archive engines upstream a MySQL database, jointly the CSS Client operator interface.

The software architecture used for tests, illustrated in figure 6, shows that the load is balanced between two servers: one for insertions and one for reads.

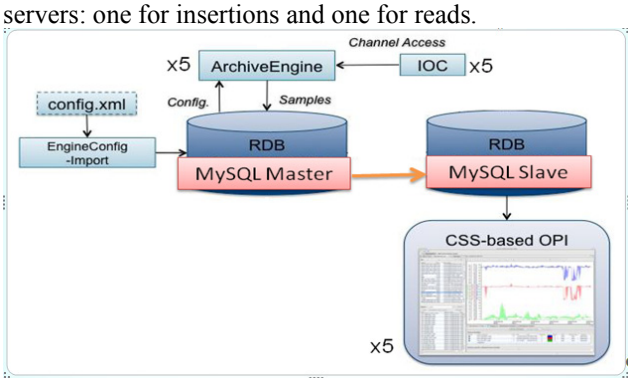


Figure 10: Test architecture for archiving.

The writing tests, presented in Table 1, were performed in September 2011.

The first results showed that we needed to update the MySQL JDBC driver and settle the **rewriteBatchedStatement** property to true to enable bulk insert and a huge performance gain.

Table 1 : Archiving Test

Number of IOC	Number of PV/IOC	Number of Archive Engine	Number of PV/ Archive Engine	f	PV/s
5	80	5	80	10Hz	4000

The reading (CSS client) tests using MySQL partitioning started recently.

CONCLUSION & NEXT STEPS

The use of the XAL library has been very efficient. While some of the SNS applications have been reused without modification: scanning applications, virtual accelerator, launcher, etc., others have had to be adapted to fit our needs: optimisation, magnet cycling, radio-frequency cavities tuning. Specific applications were designed from scratch using the XAL library and framework: admittance limitation, alignment, profiles monitoring and equipment control. New developments are foreseen: ion source spectrometry, beam power monitoring, longitudinal emittance measurement, etc.

Most of the applications require data from the database for their configuration especially through xdxf file. Furthermore, databases are at the heart of central service like alarms or archiver.

ACKNOWLEDGEMENTS

Other contributors are: from the Ganil control group, D. Touchard (Epics environment and tools), C. Haquin (real time), P. Lermine (databases); from IRFU, D. Uriot (Tracewin), JF. Denis and F. Gougnaud (Tests at Saclay).

The basic hardware and software infrastructure and the network architecture are managed and provided by the Ganil Computing Infrastructure group.

A special thanks to the OPEN XAL team for the software and support provided.

REFERENCES

[1] E. Lécorché et al. “Overview of the Spiral2 Control System progess” Icalepcs 2011.

[2] F. Gougnaud et al, “The Implementation of the Spiral2 Injector Control System”. This conference.

[3] J. Galambos, et al, “XAL the SNS application programming infrastructure” Epac 2004

[4] T. Pelaia II et al. “XAL Status” Icalepcs 2007.

[5] Open Xal : http://xaldev.sourceforge.net/.

[6] J.Galambos, et al.”Xal - U.S. Particle Accelerator School 2008” https://wiki.ornl.gov/sites/xaldocs/PHYS798X Course Material/Forms/AllItems.aspx.

[7] D. Uriot, N. Pichoff “New implement in TraceWin/Partran codes: integration in external field map”, PAC2003

THE NEW CONTROL SYSTEM FOR THE VACUUM OF ISOLDE

S. Blanchard, F. Bellorini, F.B. Bernard, E. Blanco, P. Gomes, H. Vestergard, D. Willeman

CERN, Geneva, Switzerland

Abstract

The On-Line Isotope Mass Separator (ISOLDE) is a facility dedicated to the production of radioactive ion beams for nuclear and atomic physics. From ISOLDE vacuum sectors to the pressurized exhaust gas storage tanks there are up to five stages of pumping for a total of more than one hundred pumps including turbo-molecular, cryogenic, dry, membrane and oil pumps. The ISOLDE vacuum control system is critical; the volatile radioactive elements present in the exhaust gases and the high and ultra high vacuum pressure specifications require a complex control and interlock system. This paper describes the reengineering of the control system developed using the CERN UNICOS-CPC framework. An additional challenge has been the usage of the UNICOS-CPC in a vacuum domain for the first time. The process automation provides multiple operating modes (rough pumping, bake-out, high vacuum pumping, regeneration for cryo-pumped sectors, venting, etc). The control system is composed of local controllers driven by PLC (logic, interlocks) and a SCADA application (operation, alarms monitoring and diagnostics).

INTRODUCTION

The ISOLDE Facility

The On-Line Isotope Mass Separator ISOLDE [1] is an experimental facility located at the Proton-Synchrotron Booster (PSB), CERN, dedicated to the production of a large variety of radioactive ion beams for many different experiments in the fields of nuclear and atomic physics, solid-state physics, materials science and life sciences.

Description of the Vacuum System

In terms of vacuum, the ISOLDE facility is an array of beam lines used to distribute radioactive ion beams to a number of experiments. The beam lines are separated into sectors which are often generically represented by the equipment used in that part of the machine. For example, the front-ends refer to the two separate target stations where both heating and high voltage are an issue for the vacuum system and the separators refer to the sectors where the separating magnets are located.

The new vacuum control system replaces a fifteen year old control system based on old obsolete PLC (Programmable Logic Controller) without dedicated supervisory software. The old control system became difficult to maintain and was no longer adapted to present day upgrades of the facility.

The new control system has been installed in two steps:

- In 2010, installation of the vacuum control system for ISOLDE front-ends, separators and experiments
- In 2011, installation of the vacuum control system for the post-accelerator REX-ISOLDE [2] (Radioactive beam EXperiment at ISOLDE)

ISOLDE machine requires high vacuum(1) and ultra high vacuum(2) provided by turbo-molecular and cryogenic pumps respectively. An additional bake-out system allows achieving ultra high vacuum. All the exhaust gas from the pumping systems (with volatile radioactive elements) are collected and stored into pressurized tanks. This specification is unique at CERN.

In order to optimize the gas collection and reduce maintenance, the number of primary pumps is reduced to a minimum. There is a first set of nine primary pumps for the exhaust of fifty-four turbo-molecular pumps and four cryogenic pumps; and a second set of nine primary pumps for the rough pumping of thirty-one beam-line vacuum sectors. A complex process has been developed to control this pump configuration (Fig.1).

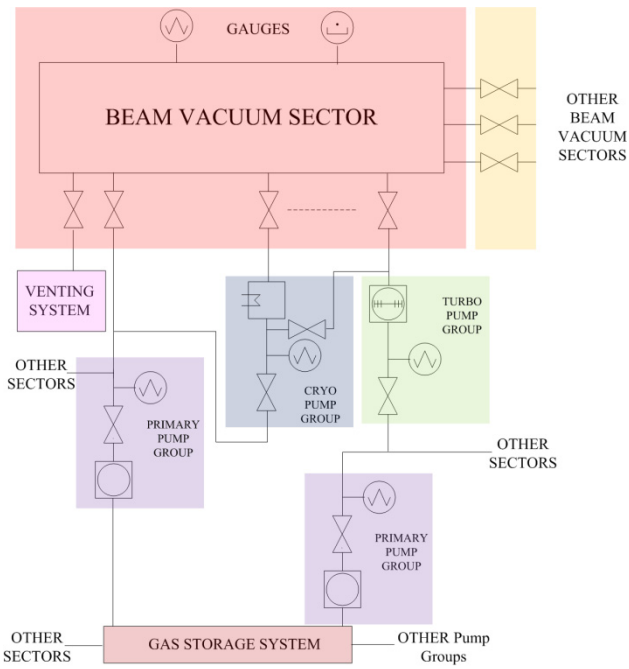


Figure 1: Typical pumps configuration.

(1) Pressures below 1.10^{-6}mbar
(2) Pressures below 1.10^{-10}mbar

CONTROL PROCESSES

The vacuum system is split in one hundred and twenty-seven processes of ten types (beam sector, turbo-molecular pump, cryogenic pump, primary pump, venting, storage, etc).

The pump process interlocks the client process (the client can be another pump or a beam-line sector). A pump process has several statuses: stop, starting, nominal (nominal pressure reached), recover (pressure increases after the pump has reached its nominal pressure), leak detection, etc. The pumping valve of the client remains closed if the pump process is not in "nominal" or "recover" status and the opening is disabled if the pump process is in "recover" status.

Some processes are multi-client (Fig.2); to avoid that two clients are pumped at the same time (retro-diffusion and contamination problems), a « token » system has been developed with low and high client priorities.

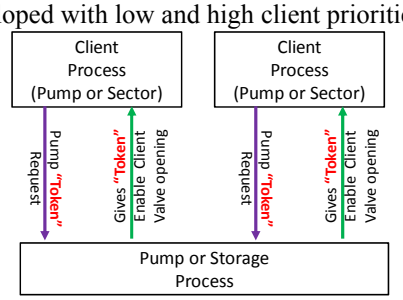


Figure 2: Pump and clients relation.

Processes are linked together with multiple interactions (Fig.3).

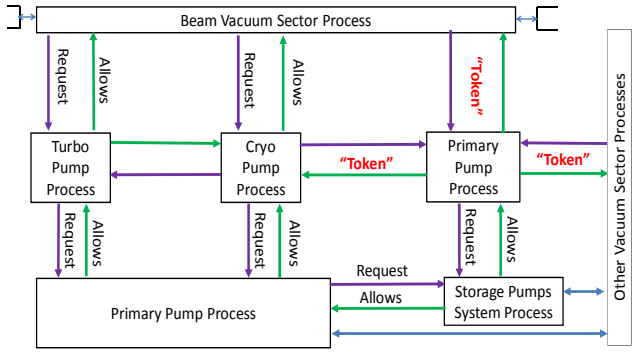


Figure 3: Processes interactions.

HARDWARE

Hardware Architecture

The hardware architecture (Fig.4) is based on Programmable Logic Controller (PLC). The vacuum control hardware is a five layer architecture: SCADA[(1)] Application, PLC, remote input/output, device controllers and field devices.

The remote input/output station and controllers are installed close to the field devices to reduce cabling.

Unfortunately, approximately a third of the vacuum equipments are located in radioactive and restricted areas. For these devices, specific radiation hard gauges, captors, local crates and cables have been installed.

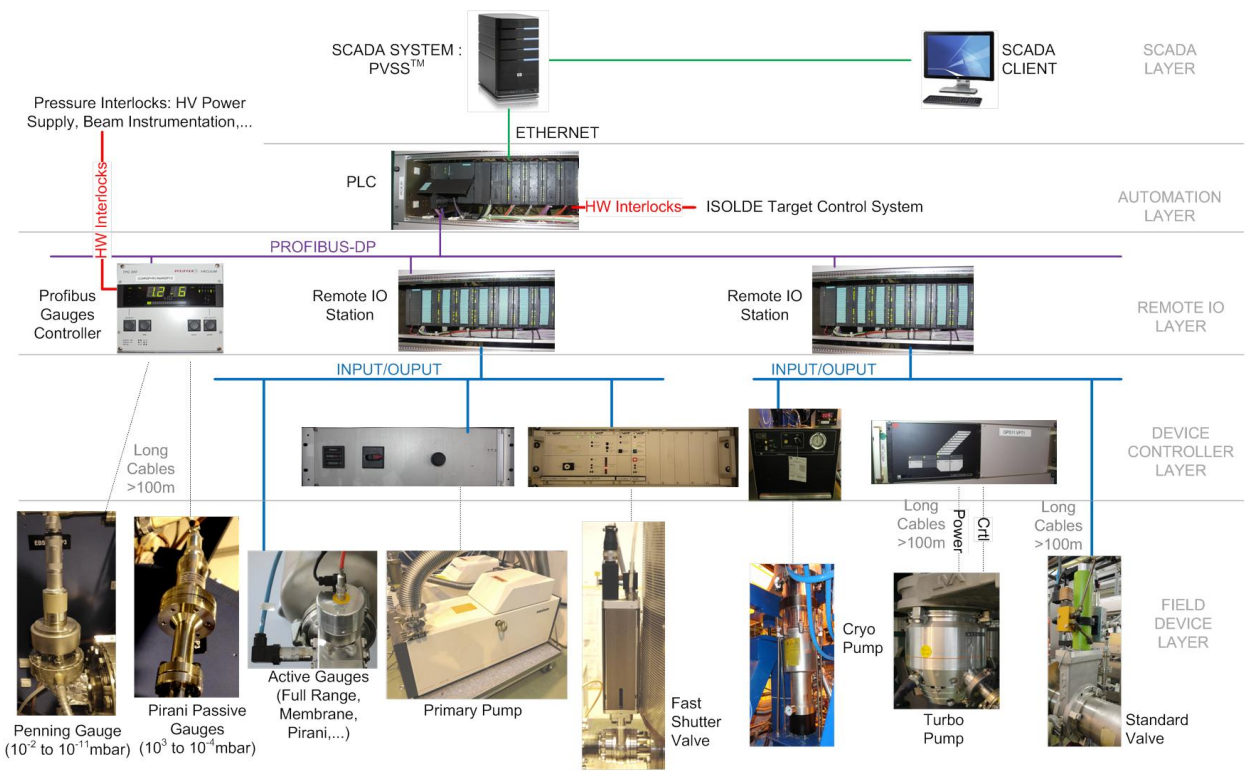


Figure 4: Hardware Architecture.

(1) Supervisory Control and Data Acquisition

Long distance cables (longer than one hundred meters) were installed and some specific cabling development for turbo-molecular pumps was required. The standard cable was replaced by two cables, one for the power of the pump motor and a second cable for the pump captors and signals.

ISOLDE and REX-ISOLDE vacuum controls are independent systems with two PLCs and two SCADA applications. The two SCADA applications and software are installed on the same Linux server.

The architecture is quite similar for the two systems and resumed by the Fig. 4.

Hardware Interlocks

The hardware interlocks are potential free relay contacts. The PLC provides hardware interlocks to other equipments like transformers, water cooling, target heaters, etc. The interlocks are the results of logic combinations of pressure levels and process status.

The gauge controllers provide fast pressure hardware interlocks to high-voltage power supplies. A PLC function has been developed for the asynchronous communication with the gauge controllers via the Fieldbus Profibus-DP™. This function allows to set, monitor and diagnose a large number of hardware interlocks.

SOFTWARE

UNICOS-CPC Framework

UNICOS (UNified Industrial COntrol System) [3] is a CERN framework to produce control applications. The UNICOS-CPC package proposes a method to design and develop complete industrial process control applications. It is based on the modelling of the process in a hierarchy of devices (e.g. I/Os, field and abstract control devices). These devices establish the base on which process engineers and programmers define the functional analysis of the process. The package is deployed both in the supervision layer (WinCC OA) and in the PLCs (Schneider and Siemens). UNICOS-CPC also provides tools to automate the instantiation of the devices in the supervision and process control layers and to generate either partially or completely the specific control logic code. The framework is flexible enough to be able to add customized devices to cover new client requirements.

Phases Sequencer

The ISOLDE vacuum control system largely uses Grafcet(1) to drive devices and activate alarms and interlocks.

The Grafcets are directly monitoring a SCADA panel (Fig.5). The operator can follow the process and see the active steps and transitions.

(1) Sequences of actions with conditions and time dependencies to achieve discrete event

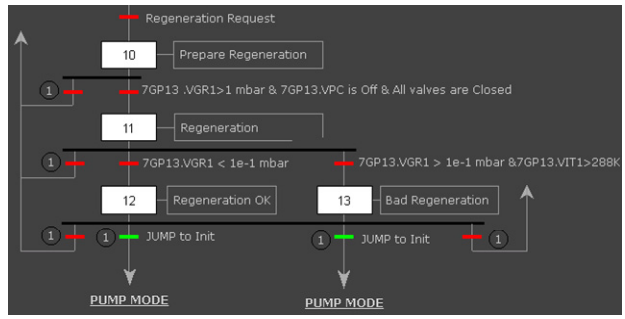


Figure 5: Part of a cryogenic pump process Grafcet (SCADA Panel).

Thirteen different Grafcets provide a fully automatic system with a large choice of operating modes: pump, leak detection, vent, bake-out, regeneration, etc. Transitions between modes can be automatic or manually requested.

Software Production

The process objects use phase sequencers to control the vacuum equipments. Physical values such as valve positions or gauge pressures are acquired through input objects and commands are set via output objects.

Phases sequencers (Grafcets) and PLC function templates are produced using the logic specifications. The "vacuum" library is composed of PLC functions developed for specific vacuum devices (analogue gauges with on/off commands, complex gauge controller devices, etc). This library was created especially for the vacuum system of ISOLDE (Fig.6).

The PLC source code is obtained using:

- Baseline code generated by UNICOS-CPC framework.
- Function templates, Grafcets and "vacuum" library.

The SCADA application is obtained using:

- Baseline application produced by the UNICOS-CPC framework.
- SCADA application objects automatically generated from the objects database and specifications.
- Fully configurable panel templates developed using the ISOLDE vacuum layout.

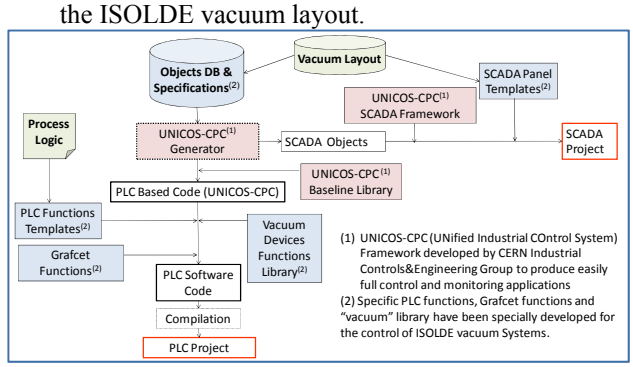


Figure 6: Software production diagram.

Software Interlocks

The project integrates phases sequencer time-out and alarm conditions to interlock devices and processes. The sector valves need to be interlocked in case of pressure increase. The PLC provides software interlocks for the sector valves, a low pressure threshold is set to disable the opening of a valve and a high pressure threshold is set for a fully interlocked valve (valve closure and inhibit opening). The SCADA panel (Fig.7) displays the detail and the source of the interlock providing a rapid and easy diagnostic.

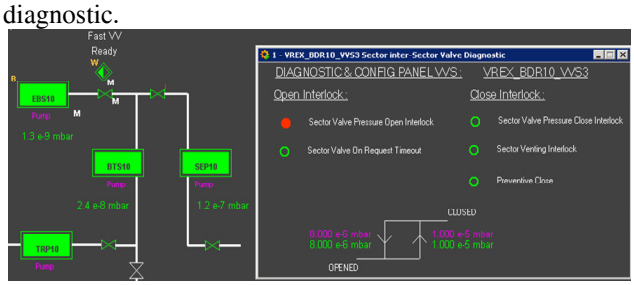


Figure 7: Sector valve interlock details panel.

SPECIAL CASES

Ultra High Vacuum: Bake-Out Control

The REX-EBIS sector is an ultra high vacuum sector and needs to be fully baked-out (Fig.8). The bake-out reduces the chamber's gas desorption and activates the in-house guetter pumps. The vacuum components are fragile and require an advanced regulation and control system. The bake-out control system is compact and mobile with PID (Proportional/Integral/Derivative) regulation, complex temperature cycles, interlocks, error management, remote control, alert and diagnostic tools. The bake-out control rack is composed of a PLC and electronic power components. This bake-out controller was developed previously by the Control Section of the VSC[(1)] Group for all the CERN accelerators[4].

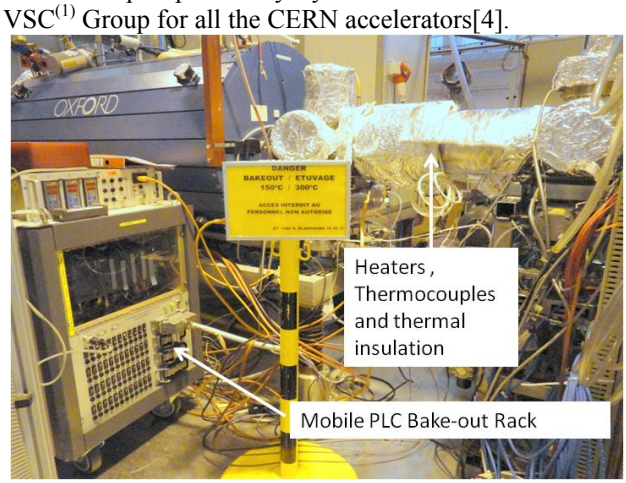


Figure 8: Bake-Out system of REX-EBIS sector.

(1) Vacuum, Surfaces and Coating Group (CERN)

Control of the Exhaust Gas Collecting and Storage System

The exhaust gas collection (Fig.9) and storage process is one of the most important processes as its failure would result in the complete shutdown of the ISOLDE vacuum system. The system is redundant and is not running permanently. Between the pump groups and the storage system, an array of decantation pipes and filters can independently accumulate gas up to a pressure of 700 mbar. The control has been designed to avoid stops or at least to prevent them. For example if the pressures in the storage tanks increase abnormally, the responsible client process is preventively stopped yet the storage system is still available for other processes.

Figure 9: Exhaust gas collecting station.

CONCLUSION

The vacuum control system of ISOLDE including REX-ISOLDE is the result of a successful CERN internal collaboration between the Control Section of the Vacuum, Surfaces and Coating Group and the Industrial Controls&Engineering Group. This system provides all the features to fully operate and monitor the vacuum system of ISOLDE. The diagnostic tools and the ability to access them remotely have considerably reduced the vacuum intervention time.

REFERENCES

[1] The ISOLDE Facility. E. Kugler. Hyperfine Interactions Vol. 129, Numbers 1-4, 23-42.
[2] The REX-ISOLDE Project. D. Habs et al. Hyperfine Interactions Vol. 129, Numbers 1-4, 43-66.
[3] E. Blanco, CERN, Geneva, Switzerland "UNICOS CPC v6: evolution", ICALEPCS'11, Grenoble, October 2011.
[4] S. Blanchard, CERN, Geneva, Switzerland "Bake-Out regulation System for the LHC", OLAVII, Warrington, March 2008; http://www.cockcroft.ac.uk/events/OLAVII/

LHC SURVEY LASER TRACKER CONTROLS RENOVATION

C. Charrondière, M. Nybo, CERN, Geneva, Switzerland

Abstract

The LHC survey laser tracker control system is based on an industrial software package (Axyz) from Leica Geosystems™ that has an interface to Visual Basic™, which we used to automate the geometric measurements for the LHC magnets. With the new version of the Leica software, this Visual Basic™ interface is no longer available and we had to redesign the interface software to adapt to a PC-DMIS server that replaced the Axyz software. As this package is no longer supported, we have taken the decision to recode the automation application in LabVIEW. This presentation describes the existing equipment, interface and application showing the reasons for our decisions to move to PC-DMIS and LabVIEW. A comparison between the new and legacy system is made

INTRODUCTION

The Large Hadron Collider (LHC) is CERN's biggest accelerator. Thousands of magnets of different varieties and sizes are used to drive the beams around the accelerator. These include 1232 dipole magnets of 15 m length, which are used to bend the beams, and 392 quadrupole magnets, each 5 to 7 m long, to focus the beams. Prior to collision, another type of magnet is used to 'squeeze' the particles closer together to increase the probability of collisions. As the required tolerances on the geometry [1] of the LHC cold masses and on the positioning of some of its components are very tight, the final steps of the assembly are assisted by 3D optical measurements. From 2001, a Visual Basic™ program written at CERN (ie: Magnet Geometric Measurement – MGM) with direct access to the command library of the Leica Geosystems™ software, called Axyz, executes every sub-routine of the measurement process [2]. To be able to use the geometric measurement program for the LHC lifetime, the upgrade of the software components was decided.

Measuring Sequence

To identify the geometry of the cold mass cold bore tubes, a motor pulls through the tubes a mechanical mole carrying a reflector whose position is measured by the laser tracker (Figure 1). Since the measurement of both tubes requires the displacement of the measuring system with respect to the cold mass, an external reference system is needed to link the different stations. This system consists of a number of fixed points, distributed around the cold mass and equipped with reflectors. The cold bore tube axes are measured followed by an on-line analysis that links the cold mass shape to the virtual theoretical shape. Once the cold mass shape is known with respect to the theoretical reference system, the cold mass components can be aligned with respect to their theoretical position. The main components to be positioned w.r.t the reference co-ordinate systems are the corrector magnets, the end covers, the cold feet pads and the extremity of the cold bore tubes. The MGM program guides the operators with instantaneous graphical representation of the measured objects and gives indications about the acceptance and the necessity of further adjustments. During the full assembly of the magnets, the relevant information about the cold mass shape and the components position are kept and used to fill in the final report.

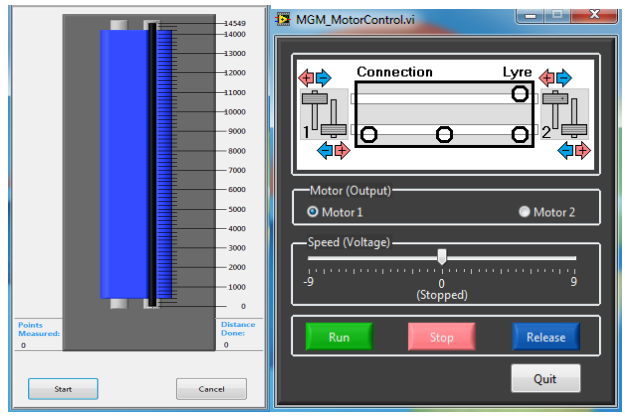


Figure 1: New motor control interface pulling the mole in the magnet cold bore tubes.

Legacy Software and Hardware

The system was running on WindowsXP, using Visual Basic™ 6.0 and Axyz software to control a LTD500 Laser Tracker. The communication between MGM and Axyz was based on OLE commands. The measurements were performed through the Laser Tracker Module (LTM) after having defined many operational parameters. The results were stored in the Core Data Module (CDM), which was accessible by the MGM for mathematical treatment of the raw data.

The list below explains the reason for an upgrade of the system:

- WindowsXP will become obsolete in the coming years.
- Visual Basic™ 6.0, is already obsolete.
- Axyz is obsolete from 2007 and not supported by Leica Geosystems™ from 2011.
- LTD500 are in good working conditions. However they are obsolete devices, thus they may need to be renewed in the coming years with more recent machines.

Software and Hardware from 2011

Following these observations, it was clear that the system had to be adapted. However the end users must not be affected and we have to keep similar GUI and work procedures (Figure 2).

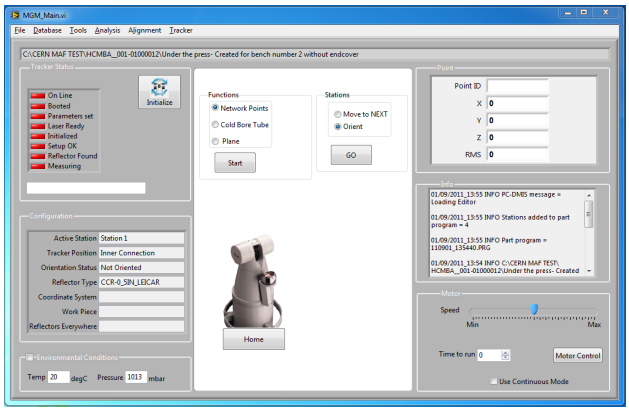


Figure 2: New MGM main panel that keeps the consistency with the previous version [3].

The most expensive tool used is the LTD500. This hardware is obsolete from the manufacturer point of view, but fulfils CERNs requirements to build and repair magnets for the LHC. However, when the need will come to buy a new laser tracker, the MGM will have to be adapted with a minimum effort. The application proposed by Leica Geosystems™ is called PC-DMIS, developed by Wilcox™, both companies being owned by Hexagon™. This application supports custom automation and can be used manually by the operator if special procedures or measurements are needed. PC-DMIS cannot be connected to the laser tracker directly, and a server (ie: emScon server) has to be added in the chain of equipment (Figure 3). The emScon server translates the request from PC-DMIS to commands understandable by the laser tracker.

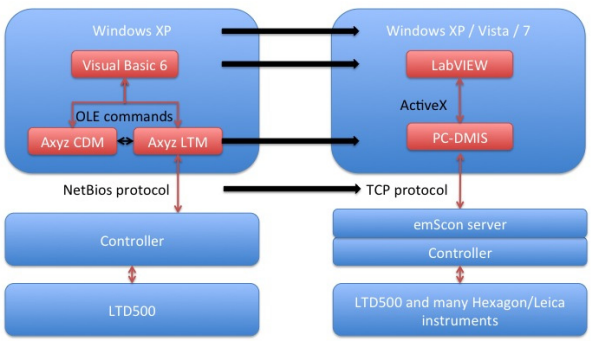


Figure 3: Software and hardware evolution.

The main difference between Axyz and PC-DMIS is that the former is point oriented while the latter is feature oriented. In Axyz the user measures a set of points and can decide later on if they describe a circle, a line …etc… In PC-DMIS the user select a feature and then measure all the points describing it. PC-DMIS is also highly graphical while Axyz is more a textual interface.

The choice was made to develop MGM using LabVIEW™, as it was the most suited to fulfil the requirements of flexibility, adaptability, quality, integration into industrial control software and light maintenance (Table 1).

Table 1: New Structure Pros and Cons

Parts	Status
Windows XP / 7	Can be upgraded to CERN standard
LabVIEW2010	Easy move from one version to another
PC-DMIS	Fully supported by Hexagon™
emScon server	Supported at least until 2017
Controller	Obsolete – can be changed
LTD500	Obsolete – can be changed

Using emScon Direct Commands

Communicating with the emScon server means sending and receiving byte-array datablocks over an asynchronous network connection. Leica Geosystems™ offers an emScon Tracker Programming Interface (TPI) for sending commands directly to the laser tracker from an external application without passing through PC-DMIS. The versatility of the emScon TPI based on standard TCP/IP allows its usage on different operating systems (Windows, Linux, UNIX, Mac…). All measuring functions are available in the emScon library. However, as PC-DMIS will use the emScon server, it is not wise to access emScon with a third party application at the same time, as this will cause conflicts at the server level. Using the emScon TPI would be way lighter for the online sequence, but features such as data storage and mathematical analysis would have to be developed in addition, while they already exist in the PC-DMIS package. For this reason emScon will be used for smaller projects not requiring heavy treatment of data. In any case, this protocol has not been chosen for MGM application.

Using PC-DMIS Commands

PC-DMIS automation gives the ability to automate repetitive tasks within a custom built application running scripts called “part programs”. Thus PC-DMIS automation is language independent and provides a list of methods, properties and events for each PC-DMIS automation object. Methods are functions that usually perform actions. Properties allow reading or writing characteristics of an object. Events are routines that get called when certain conditions are met. Performing an action like measuring a point usually involves inserting a command to the script and executing it.

One of the main advantages of using PC-DMIS is that the script describes the measurement sequence being run and stores the data in the same script. This means that an expert can open the scripts, check the raw data as well as the settings used for any measurement and re-run an offline analysis. This has proven to be highly efficient when it comes to debugging the MGM. The scripts are not only used by the expert at CERN, but also by the developers to require help from the hotline (ie: Hexagon™).

Issue Encountered

The original MGM application was developed following specifications for all the modules used to align the magnet components. From 2001 on, tuning of the application has been done on the fly and not always well documented. For this reason, details and specifications for the new version has to be sought for in the VB code. Doing the first automation of a laser tracker in 2001 using Axyz was already a challenge and the developer team had to contact Leica Geosystems™ hotline regularly for information about the hardware and the software. Now, the hardware is under the responsibility of Leica Geosystems™ and the software is under the responsibility of Wilcox™. This dual interface makes it challenging to find the right answer to a technical question involving both parts, even if Hexagon™ is managing both teams.

PC-DMIS scripting has mainly been made to automate tasks, still showing its main user interface to the operator. However, the task automation had to be transparent for the end users. A large amount of work was needed to find workarounds and to be able to play the measuring sequence in the required way.

PC-DMIS is a high level application capable of interfacing lots of different hardware and contains specialised modules for many companies. This gives thousands of objects that can be selected. Magnet manufacturing is a very narrow fields and MGM is making use of a small part of this application. For this reason, it is sometimes challenging to point out the right methods, properties or events to be used

Some functions have been added into PC-DMIS automation following requests from companies or from CERN. For instance, the function called “bundle”, used to link measurement from several laser tracker positions, was only accessible in manual operation in PC-DMIS2009 while it was a built-in feature with Axyz. From PC-DMIS2010 this feature is also accessible from the automation mode on our request.

Speeding Up Development and Maintenance

Designing the MGM with Visual Basic™ in 2001 was a real challenge. Not for the code writing itself, but the user interface and all the graphical goodies were all homemade. The mathematical treatment and graph display was taking so long to be developed in Visual Basic™ that it was already decided to develop separate modules in LabVIEW™ for these purposes. Now that LabVIEW™ has been chosen for the main application, access to the ActiveX component are just as easy as with Visual Basic™. However, the calculations, displays and data storage were already built-in functions that only needed to be arranged together.

To ease the development, a LabVIEW™ to PC-DMIS library of drivers has been created including the main functions required for the MGM. Then for each alignment module, the developer will not have to go through the entire PC-DMIS library again. If a desired function is not available, it can easily be added to perfect this LabVIEW™ to PC-DMIS palette. Each of these components is documented using the built-in online help from LabVIEW™. A homemade tool extracts this help file to create an html file, ready to be published on the web.

At the level of the application, the MGM functions are documented with a text document explaining in broad terms the action performed, and a flowchart describing in detail the action executed (Figure 4).

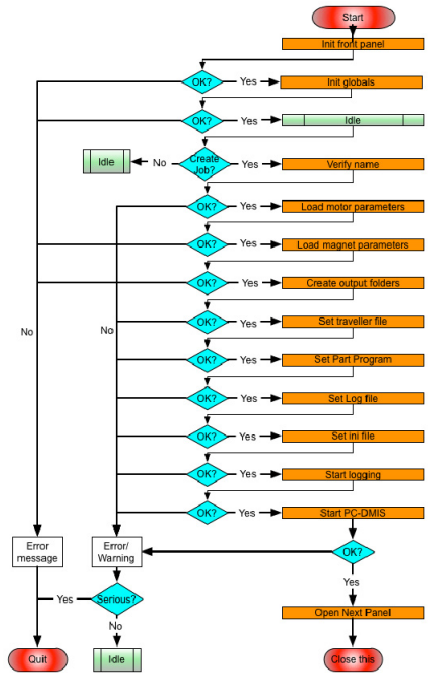


Figure 4: Flowchart for the MGM startup sequence.

A set of configuration files has also been put in place to take care of the hardware and the tolerances for the different types of magnets, as well as a database of points.

CONCLUSION

The existing MGM Visual Basic™ application has given successful results during the magnet construction for the LHC. Now the new system looks promising and the development time has been shortened a lot using LabVIEW™ and a homemade dedicated palette to PC-DMIS. The next step could be to develop a second palette for LabVIEW™ to emScon to avoid using the heavy environment of PC-DMIS for small automations.

REFERENCES

[1] M. Bajko et al., “Geometric and magnetic axis of the LHC dipole” PAC2001, Chicago, United States

[2] R. Chamizo et al., “Automation of 3D Measurements for the Final Assembly Steps of the LHC Dipole Cold Masses” IWAA 2004, Geneva, Oct 2004

[3] C. Charrondière et al., “Magnetic axis and geometric measurement for the LHC cryomagnet” ICALEPCS2003, Gyeongju, Korea, Oct 2003

THE EVOLUTION OF THE CONTROL SYSTEM FOR THE ELECTROMAGNETIC CALORIMETER OF THE COMPACT MUON SOLENOID EXPERIMENT AT THE LARGE HADRON COLLIDER*

O. Holme, D. Di Calafiori, G. Dissertori, W. Lustermann, ETH Zurich, Switzerland
S. Zelepoukine, ETH Zurich, Switzerland and University of Wisconsin-Madison, U.S.A.

Abstract

This paper discusses the evolution of the Detector Control System (DCS) designed and implemented for the Electromagnetic Calorimeter (ECAL) of the Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC) as well as the operational experience acquired during the LHC physics data taking periods of 2010 and 2011. The current implementation in terms of functionality and planned hardware upgrades are presented. Furthermore, a project for reducing the long-term software maintenance, including a year-long detailed analysis of the existing applications, is put forward and the current outcomes which have informed the design decisions for the next CMS ECAL DCS software generation are described. The main goals for the new version are to minimize external dependencies enabling smooth migration to new hardware and software platforms and to maintain the existing functionality whilst substantially reducing support and maintenance effort through homogenization, simplification and standardization of the control system software.

INTRODUCTION

The CMS ECAL DCS monitors the environmental conditions and provides control and supervision over the powering systems of the ECAL detector which forms part of the CMS experiment at CERN. The ECAL detector consists of three partitions: Barrel, Endcaps and Preshower. In total, the system comprises 972 temperature sensors, 180 humidity probes and it provides control of over 1400 bias voltage and 1000 low voltage channels. The details of the control system architecture and the operating status have been reported previously [1,2].

The CMS ECAL DCS software is built upon the supervisory control software called WinCC Open Architecture (WinCC OA) [3], formerly Prozess Visualisierungs und Steuerungs System (PVSS), from ETM professional control. The CERN developed JCOP (Joint COntrols Project) Framework [4] provides extensions to WinCC OA for the high energy physics domain and was used to facilitate the development process. The system was built over a period of more than seven years. During that time, there have been contributions from nine software developers and four hardware developers.

The CMS ECAL DCS models each piece of hardware as a separate device object. Using the JCOP Framework, the devices are organised in hierarchies, which provide views of the system in terms of off-detector hardware layout or on-detector location. One particular hierarchy includes a Finite State Machine (FSM) implemented with SMI++ [5] which provides a way to summarise the state of the whole detector and to disseminate high level actions such as power switching. The FSM is also used to sequence actions and to perform automatic protection actions when conditions of the detector deviate from the normal range.

The JCOP Framework provides several other useful features, including a Configuration Database (ConfigDB) which is used to store details of the device configuration and of groups of hardware settings, known as recipes. These features were used in parts of the control system and the application of this tool is being extended throughout the system.

The CMS ECAL DCS runs on 14 computers with all WinCC OA applications being linked together in a distributed system. The CMS ECAL DCS is integrated into the CMS DCS [6], which aggregates all sub-detector control systems.

EXPERIENCE FROM LHC DATA TAKING

Throughout the 2010 LHC data taking period and so far during 2011, the CMS ECAL DCS has shown high levels of availability and has proven to be capable of taking protective actions based on anticipated safety hazards. A 24/7 expert on-call service has ensured that requests for actions on the detector and fixing of problems are carried out promptly. All of these factors have contributed to a high level of readiness for data taking.

Statistics from October 2010 to August 2011 from CMS DCS indicate that the ECAL had a readiness of 97.8% during periods when LHC was delivering stable colliding beams. These data represent a combination of all services such as the DCS, the bias voltage, the low voltage and cooling. The predominant factors contributing to the 2.2% of non-ready time include failures of power supplies and problems with external infrastructure such as delivery of chilled water. The DCS itself was responsible for almost no downtime.

Outside LHC data taking periods, the CMS ECAL DCS continues to monitor the conditions of the detector and enables parts of the detector to be powered on and off as required. This means that the control system software must be available all of the time. Experience shows that the DCS applications have high availability and that most down time is due to planned software upgrades as well as planned and unplanned cuts of power to the DCS computers. In 2010, there was a period of degradation in a temperature monitoring system that forms part of the

*Work supported by the Swiss National Science Foundation

DCS. This was due to a single fault in the power distribution that meant that all temperature probes from one of the monitoring systems could not be read out. Work to overcome this issue is ongoing and will be discussed in the next section.

Records have been kept of the activity of the on-call service for a period of more than 12 months. The on-call activities are two-fold; detector operation during development phases and fast trouble shooting in LHC data taking periods. The data show that the frequency of trouble shooting requests has considerably dropped in the past year. This demonstrates that the software improvements, based on the feedback from the operators, combined with an increase in overall operating experience have lead to a smoother operation of the ECAL detector.

HARDWARE UPGRADES

Changes to the hardware were performed during 2011 in order to further improve the control system robustness as well as to lower its recovery time in case of faults.

The first change was a modernisation of the Controller Area Network (CAN) bus [7] interfaces. PCI-based CAN cards installed in the DCS computers have been replaced by external USB-CAN interfaces from SYS TEC Electronic GmbH [8]. These allow readout of CAN based devices via USB ports and this will save time if a faulty CAN-equipped computer needs to be replaced because there will be no need to transfer the PCI card between machines. Instead, the only step required is to connect the USB-CAN interface box and install the necessary drivers. A software library [9] from the CERN Industrial Controls & Electronics (ICE) group enabled a transparent migration between PCI-CAN and USB-CAN interfaces.

From the experience in 2010 with the fault in the temperature monitoring system, it was decided that the power distribution for this system should be improved. This involved sourcing new power supplies and constructing new power distribution blocks that allow individual switching of power to each CAN-based readout device. The system was divided into two independent parts, limiting the impact of a single fault and allowing independent power interlocks.

Further upgrades are under development. One of them aims to provide relative humidity measurements below 60% using the installed humidity probes, which is presently not possible. This system will use custom readout electronics communicating with WinCC OA via Modbus [10]. The system is being developed and manufactured at the University of Belgrade, Serbia. The Modbus data channel will be converted to Modbus TCP with a Modbus RTU-TCP adapter such that the data can be read via the standard CMS Ethernet network.

Finally, an upgrade of the Preshower bias voltage distribution system is foreseen. In the Preshower, each power supply bias voltage channel is distributed to multiple detector elements. The extension presently developed will provide greater granularity for the monitoring of currents drawn by the Preshower detector elements. While not solely a DCS issue, this additional monitoring system will require both software and hardware changes by the CMS ECAL DCS team.

SOFTWARE ANALYSIS PROJECT

Due to a reduction in the number of people working on the CMS ECAL DCS project after CMS commissioning it was essential to focus on the long term maintainability of the system.

The system functionality can be divided into monitoring applications and combined monitoring and control applications. These applications were originally assigned to individuals who then developed the software to achieve the control and monitoring goals of that application. This approach enabled the system to be delivered on time and with sufficient functionality to operate the detector.

In the current phase of the project the approach is different, with a central team of three developers being responsible for the whole DCS. This has required a large shift of information from the original developers to this new core team. During the process of centralising the detailed information about the software, certain aspects were identified that could be improved upon or standardised between applications.

As the software met all the operational requirements, the team was able to concentrate efforts on investigating ways to streamline and standardise the software. This led to the creation of the CMS ECAL DCS Software Analysis Project which took place throughout 2010. The goals of the project were to evaluate the existing software and identify ways in which the support and maintenance load over the next years of LHC operation could be reduced. This involved discovering potential simplifications, identifying areas of weakness and revisiting the original design assumptions that were embedded in the code during the original development. As the control system must run for several years, it is unavoidable that its operating environment will change. For example, operating system versions and WinCC OA versions will continue to evolve. Plans are already in place to upgrade the computers with considerably different hardware. So the analysis project considered the preparedness of the software to handle transitions such as these.

The analysis project delivered a detailed investigation of each application. This consisted of both static code analysis and dynamic analysis of the running application. The following aspects were evaluated:

- Security.
- Use of JCOP Framework and CMS DCS tools.
- Efficiency of architecture and implementation.
- Dependencies on software versions.
- Interfaces to external systems.
- FSM logic and hierarchy.
- User interface functionality and look and feel.

As an additional benefit, the analysis project produced detailed written documentation of the internal workings of the system. Previously, the documentation was focussed more on operating the software rather than how to

maintain it. As the original developers began to leave the project, detailed maintenance documentation became a high priority.

The final stage of the system analysis was the summarising and integration of the individual application analyses. This stage provided particularly good findings regarding features that could be factored out and made into generic components. Also, the compatibility amongst applications was studied to evaluate the feasibility of running multiple applications on the same machine. This would enable the number of computers to be reduced and allow more efficient use of the new generation of powerful computers.

SOFTWARE ANALYSIS OUTCOMES

Results

The process of analysing the software was successfully completed by October 2010.

The metric of lines of code (LOC) is useful to indicate the effort required to understand and maintain a code base in the long term. The total LOC of the CMS ECAL DCS is 200,000. The code required for installation and core monitoring and control functionality accounts for 40,000 LOC. The remaining 160,000 LOC are dedicated to user interface code and graphical widget configuration.

The FSM was analysed and found to have around 4,000 low-level objects to model hardware devices. There were an additional 1,000 high-level objects that summarise the device data into a hierarchical structure. A total of 61 object classes are used to instantiate all 5,000 FSM objects. It was found that a reduction in the number of classes could be achieved by combining similar classes.

The analysis detected several bugs, highlighted some deficiencies in functionality and discovered security issues that had not been identified from the experience of using the software in the CMS environment. One particular area of concern was the installation process, which ideally should be fully automatic, but in many cases was seen to fail certain steps or require additional manual configuration after installation.

In terms of consistency between components, it was noticeable that each application was developed by a different developer. Where common interfaces were required between applications, these interfaces had been defined and adhered to. Internally, architectural concepts, naming conventions, coding styles and the amount of use of the JCOP Framework varied between applications.

Planning

With the qualitative and quantitative results of the software analysis, it was possible to take informed decisions about how to reduce the maintenance effort of the system. Due to the proven stability of the existing software, it was essential that any plan for developments should not compromise the current reliability. On the other hand, there were several changes that could clearly provide benefits by reducing the complexity and heterogeneity of the software.

Two possible strategies were considered for the renewal of the CMS ECAL DCS applications. The first was to take a radical approach and re-write complete or significant parts of the existing applications. The alternative strategy was to focus on specific elements of the applications and to substitute relatively small functional blocks with newly improved elements. The former approach offered the possibility to start from scratch and re-develop a system with the advantage of the considerable knowledge acquired in the development of the existing software version. The latter piece-by-piece method provided a more cautious way to target specific problematic areas while preserving the remaining structure and code of the application.

A decision was taken to follow the piece-by-piece approach to resolve a series of issues throughout 2011. This strategy minimized the risks of altering the source code while still providing a way to significantly improve the software. With this approach, goals such as the global standardisation of function names and data object names are immediately out of scope as these would require a complete overhaul of the software. However, the benefits of a broad code cleanup did not outweigh the risks, so it was decided to keep the majority of the existing code, including its known imperfections.

The piece-by-piece approach made it possible to deploy the improvements into the production environment with minimal delay. The LHC has regular technical stops throughout the year, roughly every two months, providing opportunities to update systems without compromising LHC data taking. A laboratory setup was available for testing and validation of the software. This setup has recently been upgraded to include a computer of the new type foreseen to be introduced in CMS DCS, enabling realistic performance tests to be undertaken.

The analysis project highlighted areas in the user interface where improvements or standardisations could be made. However, operators are used to the existing screens, so only a small set of important changes were put in the software development plan.

MAJOR DEVELOPMENTS

Essential Modifications

The first priority in terms of development was to fix the security issues. These were resolved by following the CERN security team guidance [11].

The second priority of development was to ensure that the applications of CMS ECAL DCS could be installed with a completely unattended process. This is important to reduce downtime that occurs when a computer fails and the application has to be installed on a new computer. As a short term measure, work was done to improve the existing installation procedures for each application. These changes were tested in December 2010 during a long shutdown of CMS where a complete re-installation of the CMS ECAL DCS software was performed. The results showed that the installations could be completed

with almost no user interaction, although some work remained to achieve a completely unattended installation.

Configuration Database

In early 2011, emphasis was placed on moving hardware and software configuration data and hardware setting recipes into the ConfigDB. Some applications used the ConfigDB previously, but none made use of the full range of possibilities available.

One desirable feature was the facility to apply and verify hardware settings each time before switching on the low voltage and bias voltage channels. This approach ensures that all settings are correct before switching on the voltages and was previously available in the low voltage application only. In order to deploy this functionality across the whole CMS ECAL DCS, a new generic component was written that was capable of interacting with the FSM and the ConfigDB to apply recipes on demand and then verify that these have been correctly applied. In addition, the software checks to see if there is any spontaneous change of hardware settings that deviate from the stored recipe values. This component was based on development already undertaken by the CMS Tracker DCS team [12], which in turn was inspired by a tool provided by JCOP. A new generic component was designed and implemented for the CMS ECAL DCS with a view to it being used elsewhere in the CMS DCS. This component was first deployed in the CMS DCS in April 2011 and the component was integrated into all the CMS ECAL DCS applications that control low voltage and bias voltage power supplies.

Installation Mechanism

The installation procedures of the individual CMS ECAL DCS applications were all developed individually and although they all performed similar steps, the approaches were very different. From the analysis of the installation procedures, it was noted that there was a considerable mixing of code and data. As a result, each application was tuned to perform the required installation but the code to execute these steps was not generic. Based on the work performed to concentrate the configuration data in the ConfigDB, there was an opportunity to standardise and streamline the installation process.

A new data driven installation method was developed, where a single generic installation mechanism can install any application. To achieve this, a custom data object was defined that contains the data required to drive the installation process. A common piece of code interprets this data at installation time to perform all the necessary steps to setup the system. The first unattended application installation using this method was in July 2011.

SOFTWARE STATUS

The changes in the CMS ECAL DCS software so far have yielded a reduction of around 20% in the number of lines of code, when excluding the user interface. This equates to around 8,000 lines of code and has been achieved with no loss of functionality. In fact additional functionality has been added including a new application displaying the status of equipment racks.

Further developments are planned to factor out common functionality and reduce the overall software complexity. An associated reduction in LOC is anticipated. Particularly in light of the upcoming hardware changes to some CMS ECAL DCS monitoring applications, the option to make a more radical redevelopment of certain applications will be reassessed.

Changes to some applications are required to ensure that multiple applications can run together on a single computer. Performance tests are also required to test that such combinations are feasible.

CONCLUSIONS

Experience from 2010 and 2011 shows that the CMS ECAL DCS meets the required standards and has matured into an extremely reliable system. Changes have been made to the software and hardware to ensure reliability of operation and reduction of the maintenance load. The changes were implemented in a piece-by-piece way to minimize the risk associated with modifying a working system. This strategy also enabled deployment during regular technical stops of the LHC. The control system now has greater functionality even though the code base of the project has been significantly reduced. As a result of changes to both the hardware and software, the system is more robust for LHC data taking periods and better prepared for future hardware and software changes.

REFERENCES

[1] D. Di Calafiori et al., “The CMS Electromagnetic Calorimeter Detector Control System”, CHEP’2010, Taipei, Taiwan (2010).
[2] D. Di Calafiori et al., “The CMS ECAL Detector Control System”, Proceedings of ICALEPCS’2009, Kobe, Japan (2009).
[3] ETM professional control GmbH; http://www.etm.at
[4] O. Holme et al., “The JCOP Framework”, Proceedings of ICALEPCS’2005, Geneva, Switzerland, (2005).
[5] B. Franek and C. Gaspar, “SMI++ object oriented framework used for automation and error recovery in the LHC experiments”, Journal of Physics: Conference Series, vol. 219, no. 2 (2010).
[6] R. Gomez-Reino et al., “The Compact Muon Solenoid Detector Control System”, Proceedings of ICALEPCS’2009, Kobe, Japan (2009).
[7] CAN in Automation (CiA), http://www.can-cia.org
[8] SYS TEC electronic GmbH, PC-CAN Interfaces; http://www.systec-electronic.com
[9] CERN Engineering Department, SysTec Wrapper; http://cern.ch/en-ice/SysTec+Wrapper
[10] The Modbus Organisation, http://www.modbus.org
[11] CERN computer security, http://cern.ch/security
[12] L. Masetti et al., “The CMS Tracker Detector Control System”, 2008 IEEE Nuclear Science Symposium, Dresden, Germany (2008).

FIRST EXPERIENCE WITH VMWARE SERVERS AT HLS

G. Liu*, X. Bao, C. Li, J. Wang, K. Xuan, NSRL, USTC, Hefei, Anhui 230029, China.

Abstract

Hefei Light Source(HLS) is a dedicated second generation VUV light source, which was designed and constructed two decades ago. In order to improve the performance of HLS, especially getting higher brilliance and increasing the number of straight sections, an upgrade project is undergoing, accordingly the new control system is under construction. VMware vSphere 4 Enterprise Plus is used to construct the server system for HLS control system. Four DELL PowerEdge R710 rack servers and one DELL Equallogic PS6000E iSCSI SAN comprises the hardware platform. Some kinds of servers, such as file server, web server, database server, NIS servers etc. together with the softIOC applications are all integrated to this virtualization platform. The prototype of softIOC is setup and its performance is also given in this paper. High availability and flexibility are achieved with low cost.

INTRODUCTION

Hefei Light Source (HLS) is a dedicated second generation VUV light source, which was designed and constructed two decades ago. In order to improve the performance of HLS, especially getting higher brilliance and increasing the number of straight sections, an upgrade project is started from the end of 2009. This provided an opportunity to reconstruct the server system. The old server system includes several SUN workstations and some rack-mount servers providing file services, net services, database services and development environments, and these computers were all purchased 5 years ago. These years virtualization technology is applied widely, for example, it is applied successfully at CLS[1]. After evaluating the performance and cost, VMware vSphere 4 Enterprise Plus is chosen to build the new server system. Besides file services, net services, database services and development environments, softIOC applications will be also integrated to this new virtualization servers.

HARDWARE AND SOFTWARE SYSTEM

After making the decision to use virtualization technology to construct the server system, VMware vSphere 4 Enterprise Plus is chosen because it is a mature product and used most widely. The products of Dell are chosen to build the hardware platform, which consists of 4 Dell PowerEdge R710 rack servers, 1 Dell PowerEdge R210 rack server and 1 DELL Equallogic PS6000E iSCSI SAN. The figure 1 is the photo of the server system.

The technical specification of Dell PowerEdge R710 is below:

- 2 Quad-core Intel Xeon 2.53GHz processors
- 48GB DDR3 memory
- 64GB SSD
- 8 1Gigabit Ethernet ports

Figure 1: Photo of HLS server system.

The Dell Equallogic PS6000E iSCSI SAN is configured with 16*1TB SATA disk, 8*1Gigabit Ethernet ports, dual power supplies and dual controllers. It runs RAID 6 and has a hot spare drive. The usable disk space is about 11.34TB, as shown in figure 2.

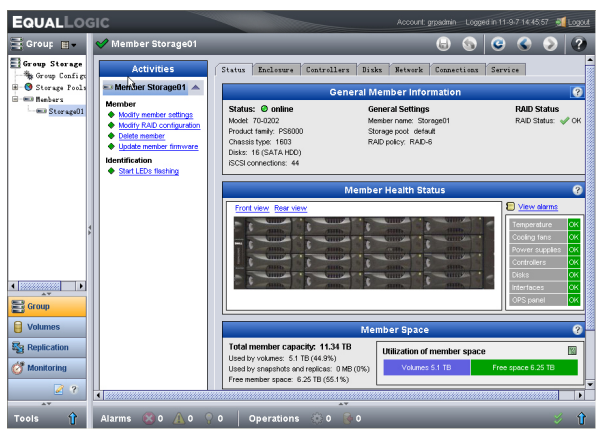


Figure 2: Management interface of Dell EQ iSCSI SAN.

The network topology is very important to the communication performance of server system. 2 switch are adopted, one for storage, the other for product, as shown in figure 3.

*gfliu@ustc.edu.cn

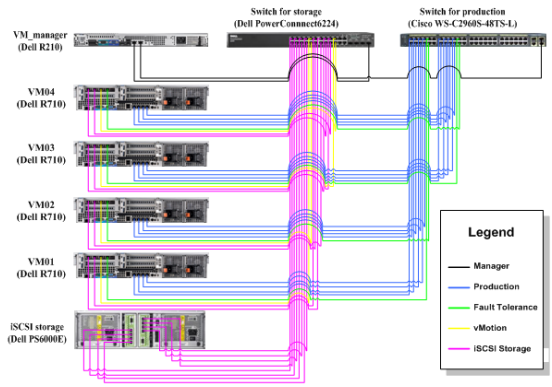


Figure 3: Network topology of HLS server system.

The virtual machine list is shown in figure 4. The virtual machine "Develop_Server" is used as the development environment, which runs CentOS5.5, "Database_Server" runs Orace10gR2, "File_Server" provides files service and svn repository, "PS_ioc" is the server for softIOC, other virtual machines are for pre-configured templates and operating system tests.

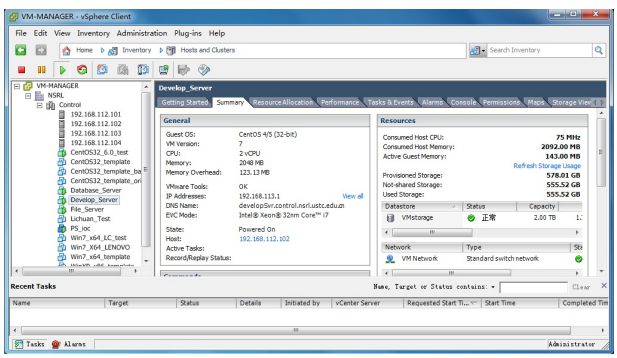


Figure 4: VMware vSphere client.

The following 4 features are related to the availability[2]:

- VMware vMotion. It enables the live migration of running virtual machines from one physical server to another with zero down time, continuous service availability, and complete transaction integrity.
- VMware Storage vMotion. It enables the migration of virtual machine files from one datastore to another without service interruption.
- VMware High Availability(HA). It provides high availability for virtual machines. If a server fails, affected virtual machines are restarted on other production servers that have spare capacity.
- VMware Fault Tolerance. When Fault Tolerance is enabled for a virtual machine, a secondary copy of the original (or primary) virtual machine is created. All actions completed on the primary virtual machine are also applied to the secondary virtual machine. If the primary virtual machine becomes unavailable, the secondary machine becomes active, providing continuous availability.

When VMware vMotion, VMware Storage vMotion and VMware Fault Tolerance are tested on the virtual machine "Develop_Server", the users of "Develop_Server" are unaware during the virtual machine migration. It looks that the service is continuous. However, VMware High Availability(HA) is related to virtual machine restarting, the service interruption is not ignorable.

THE AVAILABILITY TEST OF SOFTIOC

In order to investigate the availability of softIOC, a test system is setup, as shown in figure 5. A softIOC runs on a virtual machine, it communicates with 2 Agilent 34970A over MOXA serial device server NPort6650. A DAC board is installed on each Agilent 34970A. A SNL program running on the softIOC controls the output of each DAC board, the waveform is triangle with100ms step and is monitored by the oscilloscope Tektronix DPO4054.

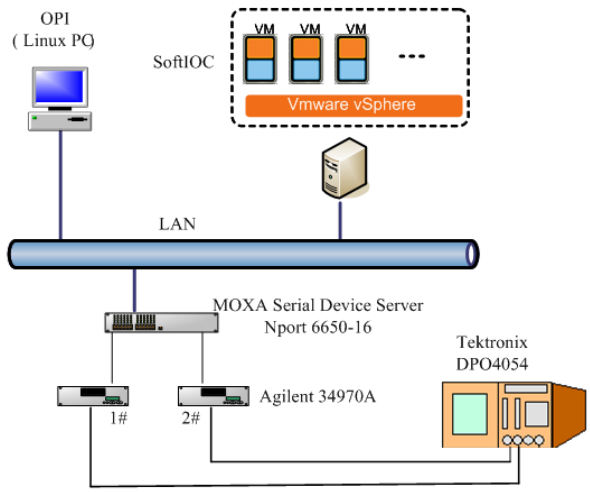


Figure 5: Hardware structure of SoftIOC test system.

The test results for VMware vMotion and VMware Fault Tolerance are shown in figure 6 and figure 7.

The figure 6 is the monitored output waveform during VMware vMotion, the interruption time is about 1600 ms. The figure 7 is the monitored output waveform during VMware Fault Tolerance, the interruption time is about 2600 ms. Although the down time is not zero, it is acceptable in our case.

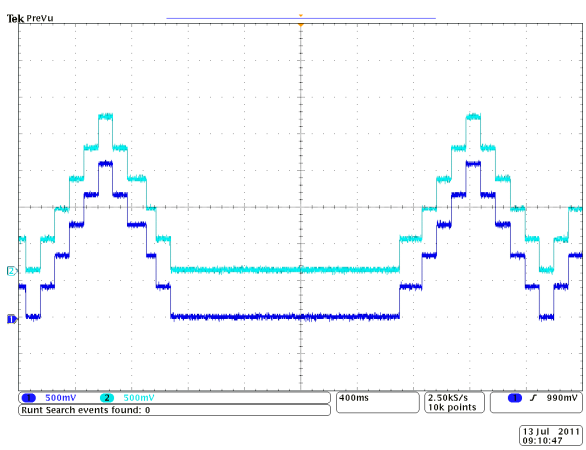


Figure 6: Output waveform during VMware vMotion.

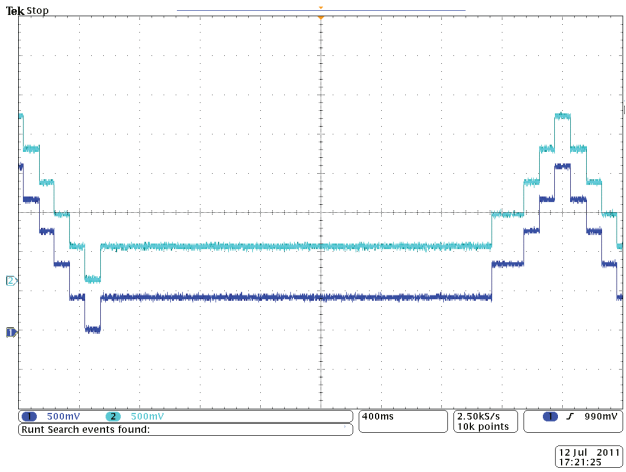


Figure 7: Output waveform during VMware Fault Tolerance.

CONCLUSION

Our first experience with VMware servers shows that the virtual infrastructure can provide a reliable, flexible, extensible and manageable environment. The down time of services during virtual machine migration is not zero, but it is acceptable at HLS. Although VMware vSphere is not cheap, cost per virtual machine will be lower when increasing the number of virtual machines.

REFERENCES

[1] G.Wright, C. Angel, C. Finlay, et al, "Migration Control Servers and Applications to Virtual Machines", ICALEPCS09, Kobe, Japan, Oct. 2009.
[2] http://www.vmware.com.

THE UPGRADED CORRECTOR CONTROL SUBSYSTEM FOR THE NUCLOTRON MAIN MAGNETIC FIELD

V.A. Andreev, V.A. Isadov, A.E. Kirichenko, S.V. Romanov, G.V. Trubnikov, V.I. Volkov,
JINR, Dubna, Moscow Region, Russia

Abstract

This report discusses a control subsystem of 40 main magnetic field correctors which is a part of the superconducting synchrotron Nuclotron Control System (NCS). The subsystem is used in static and dynamic (in dependence of the magnetic field value) modes. Development of the subsystem is performed within the bounds of the Nuclotron-NICA project.

Principles of digital (PROFIBUS-FLNR RS-485 protocol) and analog control of the correctors' power supplies, current monitoring, remote control of the subsystem via IP network, are also presented. The first results of the subsystem commissioning are given.

INTRODUCTION

The superconducting synchrotron Nuclotron [1] is one of the JINR basic facilities. It is intended to produce beams of charged ions (nuclei), protons and polarized deuterons with energies up to 6 GeV per nucleon.

The existing magnetic structure of the Nuclotron consists of 8 superperiods. Each superperiod includes 3 Regular FODO-periods and 1 period with a long drift space between the lenses. A regular FODO-period consists of one focusing and one defocusing quadrupole lenses each 0.45 m long, four dipole magnets each 1.5 m long and two free spaces for multipole correctors and diagnostics equipment placement. A long drift space is occupied by injection and extraction magnets, the internal target and etc.

The particle orbit is deformed in the real magnetic structure because of various kinds of perturbations which affect negatively on the accelerator complex operation. The orbit correction system (OCS) brings compensative perturbations of the same kind in the magnetic structure of the accelerator. The purpose of the OCS is to correct the equilibrium orbit and provide its stable position during acceleration process.The system consists of 40 correcting multipole superconducting magnets (21 horizontal and 19 vertical).

The control subsystem of the OCS is integrated to the Nuclotron Control System [2]. It gives the accelerator operators the opportunity of wide and flexible control of the correction system from the OCS console as well as from any authorized workstation of the Nuclotron LAN [3].

THE CONTROL SUBSYSTEM STRUCTURE

The common structure of the corrector control subsystem is illustrated in Fig. 1.

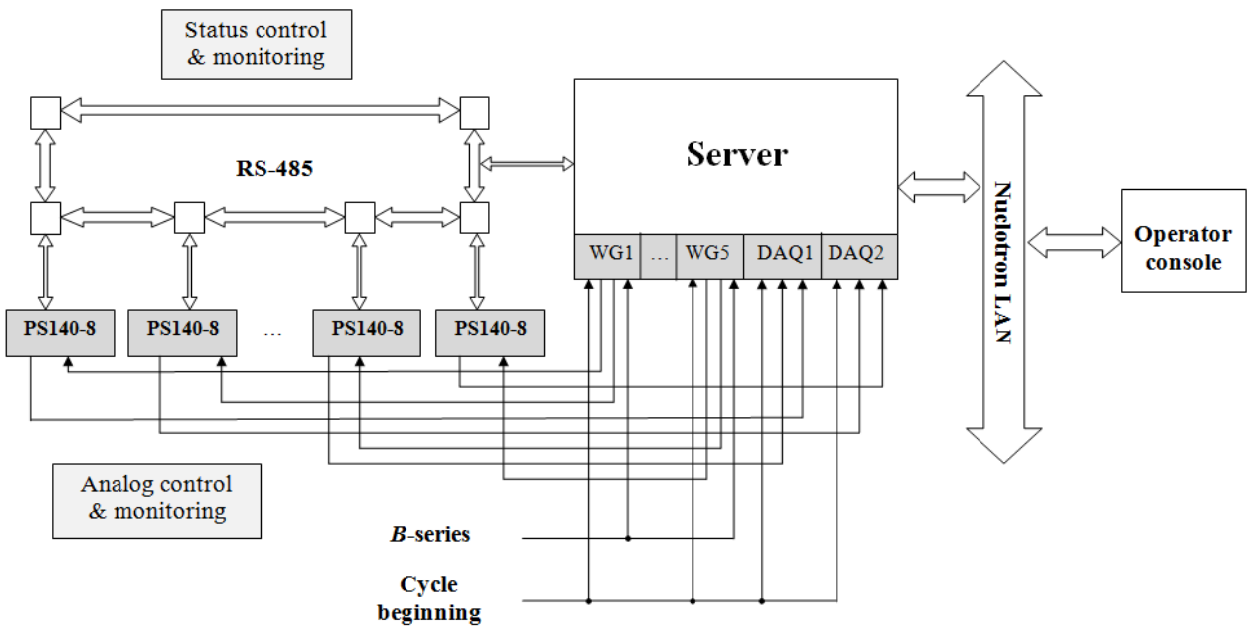


Figure 1: The control subsystem structure scheme. WG means waveform generator, DAQ – data acquisition board.

Power Supplies

The correcting multipoles are supplied by PS140-8 – the sources of current which were specially designed to supply electromagnetic elements of accelerators by «EVPÚ a.s.» (Slovakia). These sources replaced the out-of-date previous ones which had the insufficient output current range and stability. The PS140-8 power supply is a transistor converter serving as a source of DC current for R-L load with high stability of time and thermal conditions. The output current is regulated within the range of 0 to 140 A by a corresponding input signal between 0 and 10 V. The power supplies can work in a static mode as well as in a dynamic one with the current change rate from 0 to 100 A/s.

As it is mentioned above the PS140-8 output current can be controlled remotely by the external voltage reference signal. Such commands as load switching on and off, checking of the power supply status, changing of the load current direction and so on, can be carried out remotely via communication channel RS-485 (protocol – PROFIBUS-FLNR).

The Subsystem Server

The main part of the control subsystem is a rack-mounted industrial PC. It performs all interactions with power supplies such as generation of reference signals, monitoring of the power supply output current and status controlling. The server also provides the remote control of the subsystem via IP-Network (see The Control Subsystem Software section below).

Waveform Generators

To produce a reference signal of the output current we have used five National Instruments NI6733 PCI boards installed in the server backplane. These boards provide eight 16-bit resolution analog output channels with the update rate up to 1 MS/s. As it is mentioned above there are two modes of the correction system operation: static and dynamic. In the static mode the output power supply current is permanent during acceleration. If it is necessary to change the output current value the NI6733 module generates the corresponding gradual transition of the voltage. In the dynamic mode the power supply current depends on the instant value of the main magnetic field. So the image of this relation function is written to the waveform generator memory. The Nuclotron B-timer is used as an external clock source for the NI6733. Every B-timer pulse corresponds to magnetic field changing by 10^{-4} T (repetition rate is up to 10kHz).Also the module is triggered by a digital signal of “acceleration cycle beginning” in this case.

Data Acquisition Modules

For monitoring of the power supply output current two National Instruments NI628433 PCI boards are used. These modules are also installed in the subsystem server backplane. The NI6284 DAQ board provides 32/16 single-ended/differential analog input channel with an 18-bit resolution. In the dynamic mode these modules are triggered by the “acceleration cycle beginning” pulse as well as waveform generators. In the static mode DAQ boards work continuously writing the acquired waveforms to the server memory every 4 seconds.

RS-485 Communications

For status operations the subsystem server interacts with PS140-8 via RS-485 interface forming a ring topology. This communication runs on 115200 baud speed. The MOXA CP-132 PCI board provides a RS-485 interface on the server side. Power supplies are grouped on ten boxes placed along the accelerator ring. Links between the boxes are made with multi-mode fiber optics. The conversion between the fiber and serial interfaces is provided by the MOXA TCF-142-M module.

Network and Remote Station

The Nuclotron Control Room is placed 400m away from the accelerator hall where the power supplies and subsystem server are located. So it is necessary to have a remote control of the subsystem. The remote control is carried out via 1 GbE Nuclotron LAN (which is a subnet of the LHEP/JINR network). The operator can access the subsystem server from any authorized workstation connected to the LAN and running the special developed client software.

THE CONTROL SUBSYSTEM SOFTWARE

The software complex of the OCS control subsystem consists of two parts: application running on the subsystem server and client application running on operator workstations.

Server Software

The application running on the server computer performs the following main tasks: power supply status control, power supply output current setting and monitoring, data presentation and remote control providing.

To monitor the state of the PS140-8, the server starts separate thread for scanning all the power supplies in a specified address range along the RS-485 network. The thread requests the detailed status from every detected power supply and displays it on the corresponding panels of the server GUI (see Fig. 2).

One can see that the operator can give a command to power supplies to turn the load on and off, change the output current polarity and reset the error signal. It is also possible to set the output current value (in the dynamic mode it means that the current corresponds to the main magnetic field at the beam injection level). It should be noted that the waveform generators produce a reference signal for the power supply only if its load is on.

The acquired waveforms of the power supply output current may be displayed in the assembled diagram on the server application form (see Fig. 3).

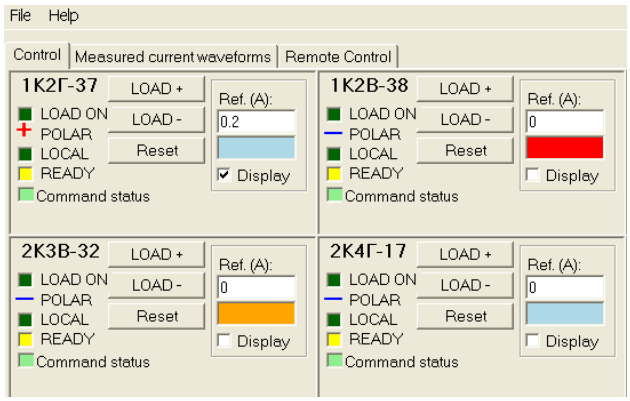


Figure 2: The PS140-8 control panels.

For the remote control realization the server starts another separate thread. It listens, checks and serves the incoming connection from the operator workstation. Any information acquired from power supply is sent to the remote client immediately.

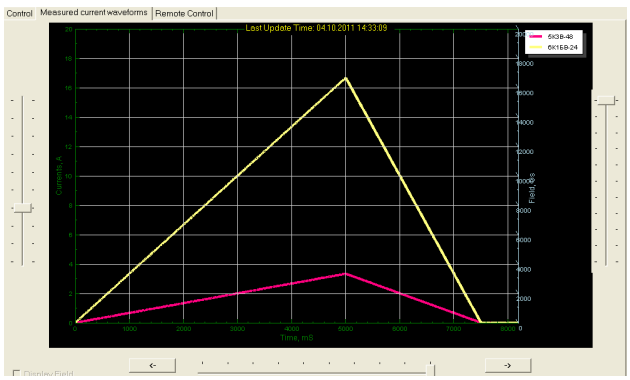


Figure 3: The assembled diagram of the acquired current waveforms.

Remote Client software

The main conception of the subsystem remote control is that the authorized user can operate corrector power supplies from the remote station just the same way as from the subsystem server console. The remote client application interface is practically identical with the server one. The server supports only one remote connection simultaneously to maintain the operation consistency. Communication between the server and remote client applications is carried out by the socket technology. The Client and Server were both developed for the .NET Framework.

FIRST RESULTS OF THE SUBSYSTEM EXPLOITATION

The new corrector control subsystem was put into operation in November, 2010. During the two latest Nuclotron runs it worked in the static mode. The subsystem has proved to be an effective instrument of the accelerator tuning. As a result of the modernized OCS exploitation the intensity of the deuteron beam (accelerated to the energy corresponding to the main magnetic field - 0.2 T) reached $5 \cdot 10^{10}$ particles [4]. It is planned to use the OCS in dynamic mode during the next Nuclotron run (November-December, 2011).

At the moment the development of the project NICA (Nuclotron based Ion Collider fAcility) [5] is carried out at JINR. The experience of the corrector control subsystem construction can be used in another part of the NICA complex – a superconducting Booster [6].

REFERENCES

[1] N. Agapov, V. Alexandrov, O. Brovko et al., “Status of the Nuclotron”, Proceedings of RuPAC-2010, Protvino, Russia.

[2] V. Andreev, E. Frolov et al., “The Nuclotron Control System”, Nuclear Electronics & ComputingXXI International Symposium (NEC'2007), Proceedings of the Symposium, Varna, Bulgaria, Sept. 10-17, 2007, Dubna,JINR, 2008, p.59-65.

[3] V. Andreev, E. Frolov et al., "Development of the Nuclotron LAN", Proceedings of 10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems, Geneva, 10 - 14 Oct 2005, PO2.065-4.

[4] A. Averichev et al., “Results of 42nd and 43rd Nuclotron Runs”. JINR, Dubna, 2011, P9-2011-72, p.8-9.

[5] NICA Conceptual Design Report, JINR, January 2008. http://www.jinr.ru/

[6] A. Butenko, N. Agapov, A.Eliseev et.al., “Design of the Nuclotron Booster in the NICA project”, Proceedings of IPAC’10, Kyoto, Japan.

SARAF BEAM LINES CONTROL SYSTEMS DESIGN

E. Reinfeld, I. Eliyahu, I. Gertz, A. Grin, A. Kreisler, A. Perry, L. Weissman,
Soreq NRC, Yavne, Israel

Abstract

A new beam line was constructed and operated for the first phase of the SARAF LINAC in order to deliver beams to various experiments. In this report concept of the beam line and different subsystems are presented. The details on the hardware and control software are given.

INTRODUCTION

The first phase of the linear accelerator of Soreq Applied Research Accelerator Facility (SARAF) has been commissioned [1]. At the moment CW proton beams at energy up to 3.5 MeV and at intensity up to 1 mA are routinely operated, while deuterons beams at energy up to 4.8 MeV and beam intensity up to 1 mA are operated only at low duty cycle. A new beam line was built in addition to the existing beam dump line, in order test new targets while protecting the superconducting LINAC from dirt coming from the target. The beams are delivered to several experiments which have broad range of requirements. The new beam line infrastructures consist of several interconnected parts: magnetic beam optics elements, beam diagnostics elements, vacuum control and machine safety system (MSS) [2].

CONTROL OF BEAM OPTICS ELEMENTS

The purpose of the magnets control is the providing the magnet currents necessary for beam transport. The control system also provides the "magnets ready" interlock for the accelerator MSS. The following magnetic elements were used for control of beam optics: 1) Two 45° dipole magnets for transporting the beam to the new beam line. 2) Five quadruples lenses (two doubles and one single) for containing the beam and preparing the required beam spot for various targets. 3) two x-y steer magnets. All the magnets except the first doublet are water cooled. The thermo switchers with 60° limits are placed on the magnet coils. Some of the magnets (dipoles, a quadruple and steers) were DANFYSIK. The two doublets were recuperated from other institutions.

The magnet control hardware located in a mountable rack located in the RF Hall (approximately 30 meters from the magnets). It is containing NI compact reconfigurable Input/Output (cRIO) platform control system and the appropriate power supplies for each magnet.

The rack includes the following elements:

1. NI AI-9201: Analog input for reading the current of the different power supplies.
2. NI AI-9264: Analog voltage Output for setting the current of the different magnets.
3. NI AI-9425: Digital Input, for indication of the operation status of the power supplies and input signals for the accelerator MSS.
4. NI AI-9476 : 32-Channel, 24 V, Sourcing Digital Output, for operation of the powers supplies.
5. GenesysTM power supplies for dipole and quadruple magnets.
6. Delta Elektronika power supplies ES030-5 with changeable polarity, for the steerer magnets.

The control system provides "magnets ready" signal for the accelerator MSS. This signal requires the conditions when all the power supplies are enables, the chain of thermo switchers is intact and the current values on the dipole magnets in the predetermined range. The power supplies could be enabled only when there is sufficient magnet cooling water flow. The beam transport condition (beam line rather than beam dump) is chosen by signal received from the accelerator control system. In the case when the beam dump line is chosen the current on the first dipole is forced to be zero in order to get "magnets ready" signal.

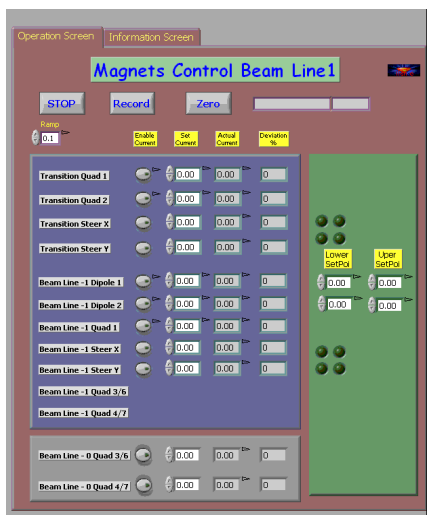


Figure 1: Beam lines magnets control system HMI.

The main screen of the magnets control application is shown in Figure 1. The application allows the user to see whether the beam line or the beam dump is chosen for beam transport. The user enables the magnet supplies and set the desirable currents. The system presents the output current and the deviation from the required current for

every magnet. The magnets are ramped to the set values with a ramping speed which also controlled by the user. For the two dipole magnets the upper and lower current threshold can be set thus enable preventing damage to the accelerator. The Information screen presents the status of thermo switchers, cooling water flow, beam line selection switch [3].

BEAM DIAGNOSTICS CONTROL

The purpose of the beam diagnostics system is beam tune and preparation beam on a target within requirements of specific experiment. The control of the beam diagnostics facilitates operation of the diagnostics elements. it also protects beam diagnostics from damage from intense beam. The beam line diagnostics consists of three of X_Y wire-profilers stations and adjustable beam collimators ("4-jaw").

Each X-Y scanner station consists of two "forks" installed in a 6-way cross. An electrically insulated 150 micron thick tungsten wire is stretched between the ends of the fork. Both ends of the wire are connected to electrical feedthroughs, so the resistance of the wire can be tested externally. The forks are moved vertically and horizontally perpendicular to the beam axis using linear motion mechanisms (Huntington) and stepping motors (NI, NEMA 23 T21NRLC). On each motion axis there are two limit switches, aimed to restrict the fork movement and thus to avoid damage to the system. The motion of the vertical and horizontal scanners is not performed on the same plane, thus, eliminating the risk of mechanical collision. After introduction into the beam, electrical charge collected on the wire is converted into voltage signal using a 1kΩ resistor. The dependence of the wire signal on its position yields information on the beam profile. These wire scanners can work only in pulsed beam (diagnostic mode) at a relatively low duty cycle. To synchronize the signal from the wires the master accelerator trigger was used as a trigger for digitizer. The wire scanners can't be used for CW beam operation (normal mode).

The control system hardware is installed in another 19' rack mountable in the RF Hall. The system is composed of several subsystems:

1. PXI chassis – with real time integrated control 8108, 2.53 GHz dual core.
2. Motion controller PXI 7350.which provides a fully-programmable motion control for the six independent axes of motion.
3. UMI 7774 for connection of the stepper servo drives (feedback and digital I/O) to the motion controllers.
4. P70360 stepper driver reading the signals from the UMI and provides the desired current to the stepper motors.
5. Digitizer PXI 5105 60MHz with eight simultaneously sampled channels.
6. The PXI 6229 digital I/O card that is used for general purpose.
7. The Motion controller (National Electrostatic Corp) provides control of the Slit Motor Drives

The control software project includes two parts: the RT controller software (PXI) and the Host computer (PC). The host computer Vi's handle the graphical user interface (GUI) and data record, while the RT controller loaded to the PXI includes the main operation Vi's and the Shred Variable Library (SV). This configuration of the VI's makes it possible to run the main program on the PXI without the need to operate the host computer, making it much more reliable and robust.

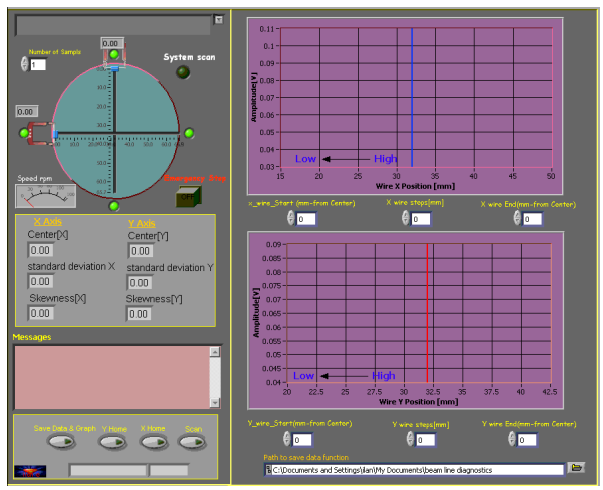


Figure 2: Diagnostics control system HMI.

Wire scanners control application is shown in Figure 2- The right part of the screen presents the scan results as an X-Y graph and the scan parameters. The profile characteristics: centroid, standard deviation and skewness, are presented in the left middle screen. The upper left corner presents the current wires positions, their velocity and operation status (scan mode). Additionally there is a message field for error notifications and scan status. At the bottom left corner the user sets the operation mode. The application allows the user:

1. Scan process: the system starts the scan process based on the defined parameters (range, steps).
2. Returns the wire to the home "out of beam" position.
3. Save data file and plot picture
4. Emergency stop of the profiler movement.

In order to prevent damage to the system there are several software protection features:

In the scan process the system will not start if both limits (Home and Forward) are inactive. b) Scan process will be stopped if the home limit in the perpendicular second axis is inactive. c) If the requested scan distance is greater than the distance between the limits, then the program will correct it automatically with a corresponding message. The "diagnostics out" signal based on the status of the profiler limits is sent to the accelerator MSS. The accelerator cannot operate in full power mode if the profilers are inserted in the beam pipe.

The High Power Beam Slits "4Jaw Collimator"

The National Electrostatic Corp 4Jaw Collimator is located at the end of the beam line. Its purpose is to scrap the beam tails and eliminate possibility of irradiation of the target vicinity. The system consists of four cylindrical elements which intercept and define a partial beam which passes between them. The elements are constructed of tantalum, and are directly cooled, by de-mineralized water. Each cylindrical element has a power dissipation >1000 Watts with highest working temperature up to 200°C. Position of each element can be varied in the range of 17 mm which defined by two limit switchers. The current from on each element can be read only for the case of CW beam. The data slit controller (from National Electrostatic Corp) handles the movement of the four slits.

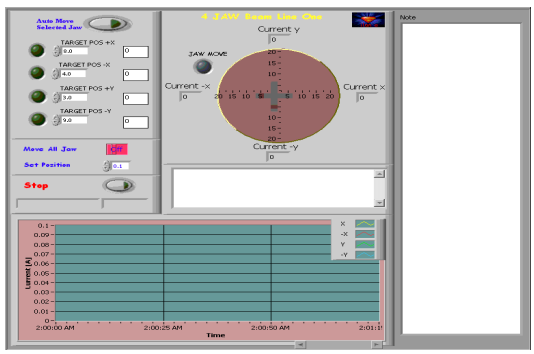


Figure 3: 4Jaws collimator control system HMI.

The main screen (Figure 3) enables the user to control the four slits. The right side of the screen presents the current location of each slit and the current evolving on it (mA). The left side enables the user to select which slit to move (by pressing the green button), and to which location (0-16mm). Moreover the user can choose to move all the slits simultaneously to the same location by using the command button “Move All”. The centre of the screen presents the messages centre. The evolving current (mA) is shown in a graph format at the lower part of the screen.

BEAM LINES VACUUM CONTROL SYSTEM OVERVIEW

SARAF vacuum control system was designed by ACCEL and commissioned at SARAF. The system was delivered to SARAF team to control the accelerator's seven vacuum sections. The control system was able to display the vacuum reading, operate gauges (open and close commands) and start the pumps.

When the first beam lines were constructed, they required additional control mechanisms and a new design concept was tailored to control the new vacuum sections. In addition, the older accelerator vacuum control system and the new beam lines control systems were unified into one control system which standardizes the display and operation, and creates common principles on which the control system operates (Figure 4) [4].

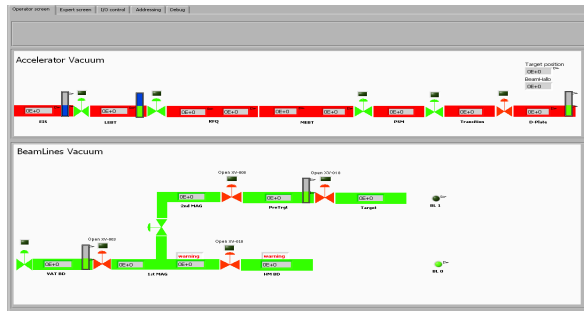


Figure 4: Vacuum control system HMI.

Principles of Design

The older accelerator control system is a simple system on which the raw information is displayed on screen with minimal processing (only scaling is performed on the vacuum readings to fit them to the various sensor types). Operation commands and statuses are also directly connected to the HMI and so no processing is done. Much unnecessary data was displayed on screen, as well as some dysfunctional controls which created an unfriendly system to use.

To correct these issues, a new display method was chosen, which minimizes the operators need to read and process the vacuum readings. The system was reorganized to display current status of the system according to a simple colour code, where green status is good (and open in case of gauge display), red status is bad (or closed), yellow status is hardware error and gray status is initial state. In normal mode, where all vacuum readings are correct and no set points were triggered, the display is all green.

The beam lines control system includes a control software element for every physical hardware element. This allows the system to be parametric, as each new addition of hardware such as gauges, pumps and sections requires a similar addition of compatible control element and with relative ease (Figure 5).

The control elements are all similar, and are configured by a central configuration method which is located at the HMI expert screen. The complex system which was implemented in the older accelerator vacuum system was reduced to a collection of control elements running in parallel and is reflective of the vacuum hardware elements installed in the facility.

Any information or feature which is not necessary and/or required during normal operation of the system was removed from the main operator HMI control panel and placed in the HMI expert screen. All configuration methods and features were placed in configuration tabs available to expert operators only.

Additional Features

As the system now shows a full view of the vacuum system, it allows the addition of new control displays such as beam blockers which are now displayed in the HMI operator screen. Additional layers of information can be added or removed from the HMI according to the operators requirements.

The control system processes the information retrieved by the control channels and indicates hardware errors in addition to good or bad operation status. Processing the current state of the vacuum system in real time also allows the implementation of machine safety procedures, so that the system can prevent the operator from making basic mistakes such as open gauges when they should not be opened.

The application includes an alarm scheme which warns the user from various situations such as gauge override command localized hardware issues.

Implementation

The system is implemented using Labview and National Instruments data acquisition (DAQ) devices. The parallel characteristics of the system require methods to prevent race conditions especially at the data out points. The parallel functions are managed by a top layer, which handles the I/O channels of the DAQ device. As there is only one DAQ device handling the I/O it has to be controlled via one connection from the control system. The management layer of the application communicate with the control elements, and convey the current snap shot of the output signals to the DAQ device. In addition, the input signals are read by the same layer and distributed to the control elements running at the system.

To prevent access to the shared resource which is the output signals snap shot, the control elements use semaphores to lock access to the output table and update its relevant signals. When done updating, the control element releases the semaphore allowing for another element to update the table and so forth.

To allow fast addition of control elements, the system is configurable. Each element is controlled by its function which is instanced when the application is started and runs continuously. As each control element acts the same, the point of configuration is when it is instanced. This configuration is done via configuration tables, so that the user has full flexibility to configure the system without changing the source code apart from instancing a new control element.

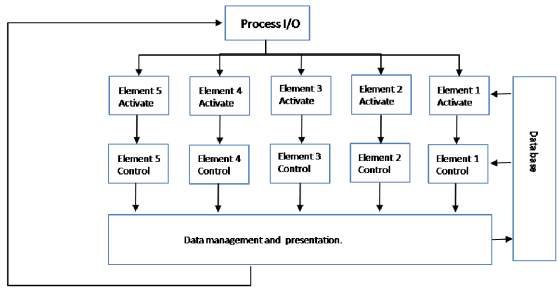


Figure 5: Vacuum control system block diagram.

SUMMARY

The SARAF beam lines control systems consists of various hardware design by the SARAF engineering team, and are currently in various stages of operation and delivery.

REFERENCES

[1] http://www.soreq.gov.il/default_EN.asp.

[2] L. Weissman, D. Berkovits, I. Eliyahu, I. Gertz, A. Grin, S. Halfon, G. Lampert, I. Mardor, A. Nagler, A. Perry, J. Rodnizki, K. Dunkel, M. Pekeler, C. Piel, P. Vom Stein, A. Bechtold, Proceedings of LINAC 2010, WE102, Tsukuba, Japan (2010)

[3] I. Mardor et al., "The operation Concept of SARAF", LINAC'06, Knoxvill, August 2006, MOP033, p. 109 (2006).

[4] I. Gertz et al., "Status of the SARAF Control System", ICALEPCS2009, Kobe, October 2009, TUP109 (2009)

DEEP-SEATED CANCER TREATMENT SPOT-SCANNING CONTROL SYSTEM

W. Zhang, S. An, G.H. Li, W.F. Liu, W.M. Qiao, Y.P. Wang, F. Yang,
IMP, LAN Zhou 730000, P.R. China

Abstract

System is mainly composed of hardware, the data for a given waveform scanning power supply controller, dose-controlled counting cards, and event generator system. Software consists of the following components: generating tumor shape and the corresponding waveform data system, waveform controller (ARM and DSP) program, counting cards FPGA procedures, event and data synchronization for transmission COM program.

INTRODUCTION

HIRFL cooling storage ring (HIRFL-CSR) is a national big science project. It consists of injector (SFC), beam transport line, the main loop (CSRm), experimental ring (CSRe) and connecting to the main ring and the ring of secondary radioactive beam experimental line (RIBLL2) and other components, the total length of about 500m [1]. Figure 1 show an Overall Layout of HIRFL-CSR. It is the heavy ion beam acceleration, accumulation, cooling, storage, and then proceeds to internal / external target experiments and high-resolution particle detection of large-scale experimental apparatus. It's built and put into operation for the physical and related research, experiments provide a good platform. Since 1995, Institute of Modern Physics, Lanzhou heavy ion accelerator using cooling storage ring (HIRFL-CSR) to provide heavy ion beams to carry out the biological effects of heavy ion beam irradiation and its mechanism, has laid a heavy ion beam cancer treatment technology base[4]. In 2009, we built on the HIRFL-CSR depth treatment of the terminal used for deep tumors. The control system is to realize the point of scanning mode in deep tumors.spot-scanning system of deep tumors is to solve the scan uniformity and conformal requirements automatically carried out.

SYSTEM STRUCTURE

The system mainly consists of hardware and software. Hardware to complete the scan waveforms output, measured data channel count and the establishment of the function. Software for data processing, transmission and analysis capabilities. System components as shown in Figure 2.

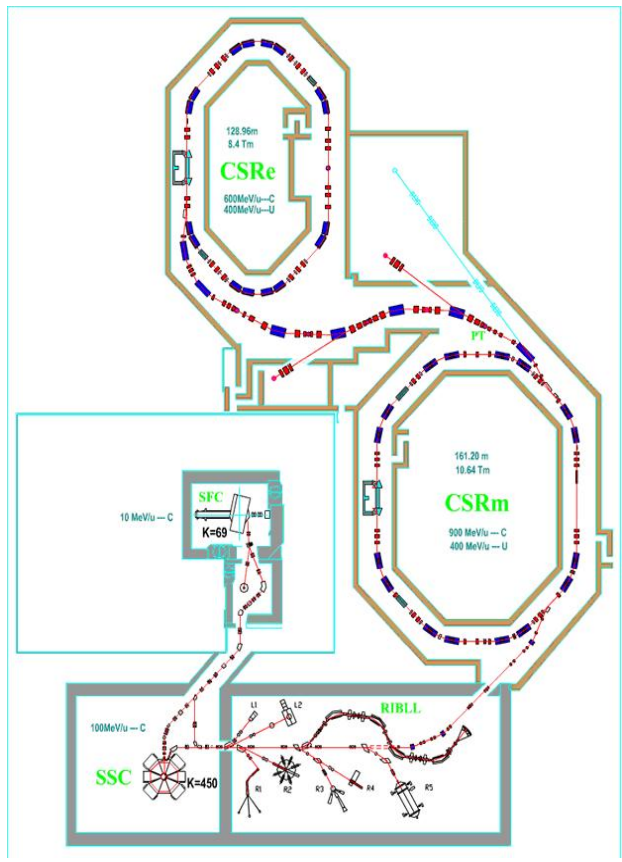


Figure 1: Overall Layout of HIRFL-CSR.

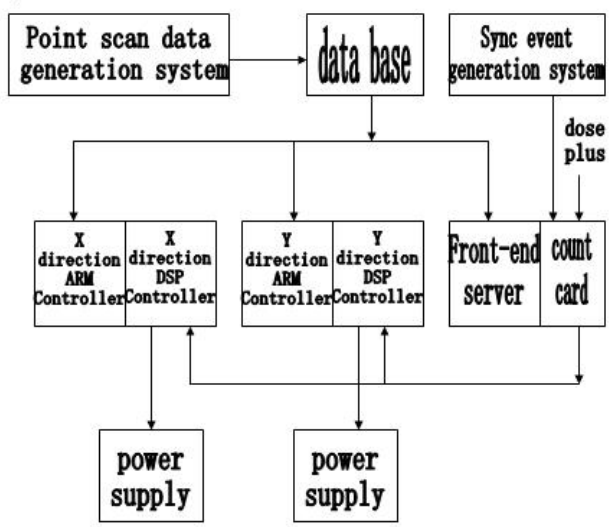


Figure 2: System components.

HARDWARE COMPOSITION

The hardware is composed of the data waveform controller hardware, counters and event generator system. The data waveform controller designed by our own is by the ARM controller and DSP controller composition. ARM controller is responsible for contact with the database. It requires the output of the waveform data from the database, then put it into a format defined by DSP controller memory. DSP controller is responsible for the corresponding input data processing, and the corresponding required output waveform. Counting cards is by our own design of the FPGA controller. It is the main achievement of the input dose pulse count and pulse required to issue hop and protection signals. Event generator system is also designed by our own FPGA controller. It is mainly to achieve our own definition of the output pulses formed by a series of examples, for the synchronous control system equipment.

SOFTWARE COMPONENT

There are three main parts. The first part is the bottom of the control equipment procedures, including ARM controller, DSP controllers, counters, and event generator system program. The second part is to generate waveform data corresponding to the tumor shape and the program. The program is the main interface operating procedures in deep spot-scan. The interface shown in Figure 3[2]. The third part is the transfer case and the COM data synchronization process [3]. It implements the DSP controller output waveform database and data synchronization, Provide the conditions for the active transformation of energy. The fourth part is the output signal monitoring procedures. It implements the DSP controller, real-time monitoring of the output waveform. The monitoring interface is shown in Figure 4 [2].

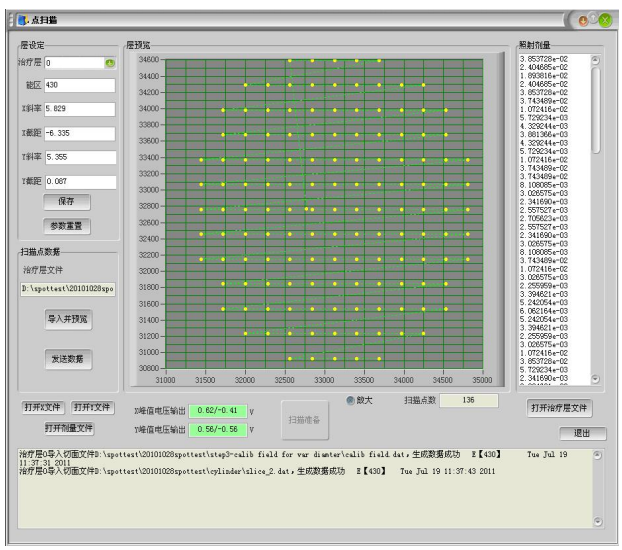


Figure 3: Interface operating in deep spot-scan.

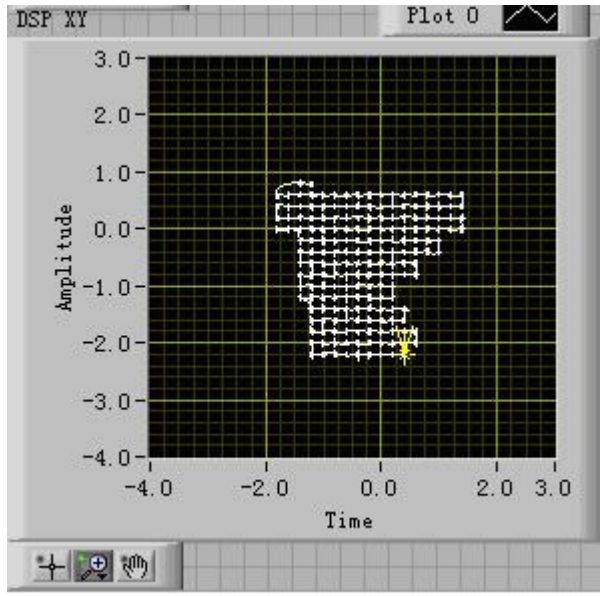


Figure 4: Output signal monitoring interface of DSP controller.

WORK PROCESS

The shape and the points according to the tumor dose requirements，Prepared using our own point of scan data generating system generates the appropriate power supply output data and counting card data。And the data pass the database, when the database receives the data, it will simultaneously transmit data to the corresponding ARM and the front-end servers。The ARM will then pass the data in DSP, waiting for the trigger pulse to the output after the。At the same time front-end server will count card data transmitted through the PXI bus, counting cards. When the count received by sync event generator card issuing an example of the start of scan, the count of cards received by the dose given detector pulse began to count, When the count to the count stored when the card count to send a pulse to the DSP, DSP receives a pulse from the current output value will be a smooth transition to the next treatment point of X, Y output value, Counting cards at a time to continue to receive from the dose given detector pulse counting, counting to then send a pulse to the DSP, this loop until the scan is complete, a layer of points. Repeat the above steps for different layers can be. Through the above process can be achieved on the different shapes of the spot scanning treatment of cancer.

CONCLUSION

In January and May 2010, we conducted a spot scanning system, the overall test, the test results meet the design requirements, and on-site test is shown in Figure 5 and Figure 6. In July 2011, point scanning control system through the acceptance of experts.

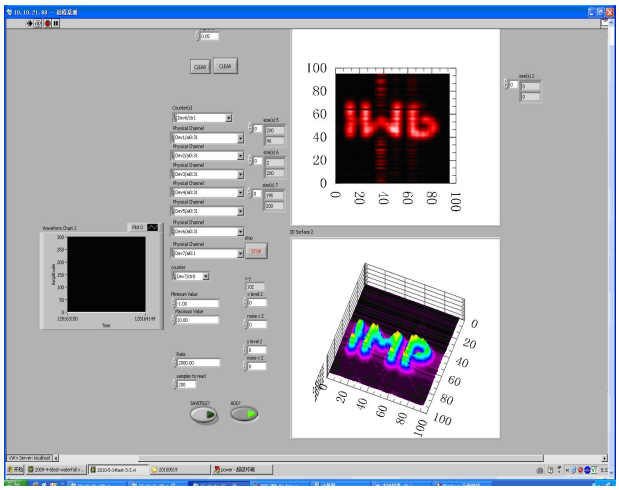


Figure 5: On-site test is shown 1.

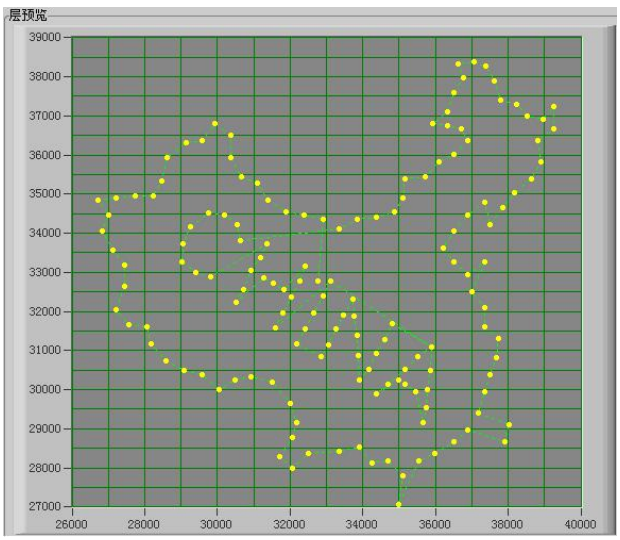


Figure 6: On-site test is shown 2.

REFERENCES

[1] Xia Jia-Wen, ZHAN Wen-Long, WEI Bao-Wen, et al. "General Design of the CSR Project in IMP", 2006, Vol.30(4)：335-343

[2] Xihui.Chen, "LabVIEW 8.20 program design", Tsinghua University Press; 2007 (In Chinese)

[3] Xu Yang, Qiao Wei-Min, Liu Wu-Fen. HIRFL-CSR "Embedded database design and implementation." Nuclear Electronics & Detection Technology. 2008, 3：590-592

[4] Xia Jiawen, Zhan Wenlong, Wei Baowen, et al. "Heavy Ion Ring Project in Lanzhou", Nuclear Physics Review, 2001, 18(3):35-38.

CONTROL OF THE SARAF HIGH INTENSITY CW PROTON BEAM TARGET SYSTEMS

I. Eliyahu, A. Arenshtam, M. Bisyakoev, D. Berkovits, I. Gertz, S. Halfon, N. Hazenshprung, D. Kijel, E. Reinfeld, I. Silverman, L. Weissman,

Soreq NRC, Yavne, Israel

Abstract

Several experiments are being prepared or conducted at the first phase of the SARAF linac. Two of these experiments are: the Liquid Lithium target and the Foil target. The present report describes the new hardware and software control systems which are currently being built for these experiments.

INTRODUCTION

A temporary beam line has been built for the 4 MeV 1 mA CW proton beam at phase I of the Soreq Applied Research Accelerator Facility (SARAF) [1,2]. Two of the experiments planned for this new beam line are the Liquid Lithium Target (LiLiT) [3,4,5] and the Foils target [6]. New hardware and software control systems were built and being tested for these experiments. The LiLiT target will be used to create an intense keV neutron source, produced via a $^{7}Li(p,n)^{7}Be$ reaction. This target was successfully tested off-line and will be installed at the accelerator beam line in the near future. The foils target is planned for irradiation experiments, examining issues of radiation damage in thin metallic foils. The control systems of both experiments include various diagnostic elements, vacuum control, temperature reading etc. These systems were designed to be modular, so that in the future new targets can be efficiently replaced.

THE LiLiT TARGET

The LiLiT target is a windowless forced liquid-lithium jet in vacuum [3,5]. Lithium is heated to ~ 200°C (above the melting temperature of 181°C). The liquid-lithium loop is designed to generate a high velocity lithium jet (10 m/s or larger). The lithium nozzle and the concave supporting wall are located inside the target vacuum chamber. An electromagnetic (EM) induction pump (figure. 1, E) drives the lithium flow from the containment tank (figure. 1, D) to the lithium nozzle. After passage through the target vacuum chamber, the lithium is collected back into the containment tank (figure 1, D). A heat exchanger, based on a separate loop of oil, circulating around the lithium tank, was designed to remove heat from impinging proton beam (up to 10 kW).

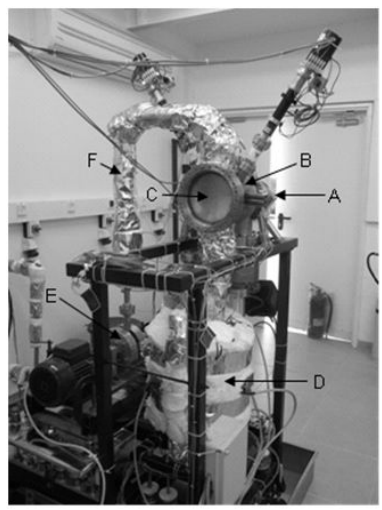


Figure 1: The LiLiT target: A-proton beam entry (will be connected to SARAF proton beam line), B-Target chamber (with lithium nozzle inside), C- neutron port, D-Lithium containment tank, E-Electromagnetic (EM) pump, F-lithium jet loop line.

A stand-alone fire resistant laboratory was built to test the liquid-lithium loop, its heat removal capabilities using an attached electron gun as a heater and to establish the safety regulations. The LiLiT control system provides full and safe control of the lithium and its cooling oil loops. The control system is based on a National Instruments compact field point (cFP) 2020 controller. The controller governs the following I/O interface modules:

1. NI-AI-111 analog current input for reading the temperatures values at various positions of the Li tank. The temperature readings are converted to current by a ConLab USD-2 universal transmitter.
2. Two NI-AI-110 modules, of analog voltage and current inputs for reading the vacuum, Li speed, temperature etc.

3. AI/O 610 analog input/output module for operation of the Li and oil flow control, linear control etc.
4. DO 401 digital outputs for Li heaters, fans, Li and oil motor etc.
5. DI 304 for monitoring of Li temperature status and alert signals.
6. TC 125 for direct temperature reading.

Additionally, the control system includes drivers for the Li and oil motors, vacuum controllers, linear motors for the diagnostic wire scan and for the beam blocker, temperature controllers, etc.

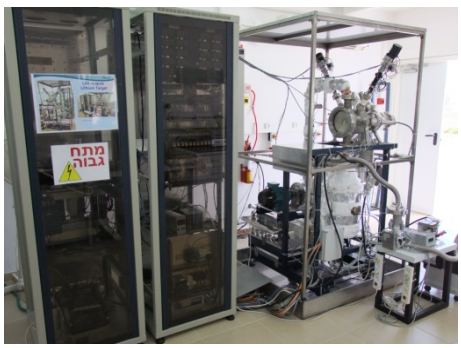


Figure 2: LiLiT target and control racks in the off-line lab

The code for the cFP controller is divided in two subsystems. The real time controller program with the critical code controls the most important parameters: the different set points (lithium speed, vacuum safety level, etc.) and the parameters of lithium heating process. The host computer program handles all the other control functions.

The Humane Machine Interface (HMI) was built to reflect as closely as possible the system structure, presenting its functionality and parameters in the most intuitively clear way (figure. 3).

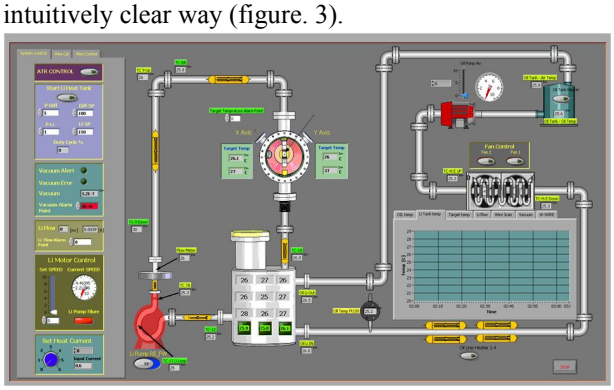


Figure 3: LiLiT control HMI

The left side of the screen enables operation of the various systems, such as the local temperature controllers, the Li tank and other heaters, the EM pump motor and wire control. Given the safety concerns associated with liquid lithium, the initial heating process of the solid Li must be performed carefully using a proportional temperature feedback algorithm. The Li loop is presented in the centre of the HMI. The diagnostics of the loop include the temperature mapping of the various locations in the loop (in the Li tank, target and pipes) and control of the EM pump. The oil loop is presented on the right side of the screen. This loop handles the operation of the oil motor, cooling fan and the oil heaters in the pipes. On the same side, the different parameters are also shown in the running plot.

Before installing the LiLiT experiment at the SARAF proton beam line, we intend to substitute the controller of the system with a new one, a cRIO which has field-programmable gate array (FPGA). This is necessary because the current controller has a relatively low performance and memory and it lacks an FPGA.

FOIL TARGET

The foil target is used to examine the effects of proton irradiation of thin foil targets. A 25 microns thick Havar foil of 20 mm diameter is irradiated by an intense proton beam. Another side of the foil is cooled by liquid metal NaK alloy, which is in contact with a water cooled stainless steel plate.

The Foil target control hardware is located in a mountable rack in the RF Hall (approximately 30 meters from the target). It contains an NI compact reconfigurable Input/Output (cRIO) 9073 control system. The cRIO includes three components: a real-time controller, a reconfigurable FPGA, and industrial I/O modules. We used the following I/O modules:

7. NI AI-9205: Analog input for reading the different devices inputs.
8. NI AI-9264: Analog voltage output for operation of the water flow and the x-y scanner.
9. NI AI-9421: Digital input used to indicate the status of the flow switch.
10. NI AI-9421: Digital output for the operation of the accelerator Machine System Safety.

The Foil control systems HMI (figure 4) is divided to two parts, the first of which operates and controls the water cooling systems (figure 4). This part presents the differential pressure gauge, temperature of the water in different locations along the pipe line, ultrasonic water flow meter, pressure gauge, water valve control and water flow switch. The second part in the HMI presents the target parameters (figure 5) such as target currents, temperatures in several locations (measurement by 4 thermocouples) and target temperature as measured by a

pyrometer sensor. The control systems enable the user to direct the pyrometer to a specific point on the target or to perform a two dimensional scan. The scan system consists of two drivers (MicroMaxTM Series 671) and of two accurate mirror positioning systems.

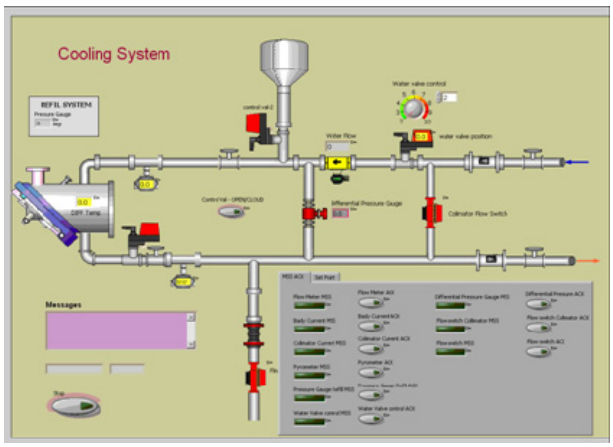


Figure 4: Foil target control system – water cooling systems.

The control system provides a "target ready" signal for the accelerator Machine Safety System (MSS). We therefore used the FPGA, which provides high reliability, high speed and determinism. The "target ready" signal is set by the chain of the following signals: water flow value, pyrometer temperature reading and flow switch status. The "target ready" signal ensures that the target cooling system is operating and that the target temperature is in the permitted level.

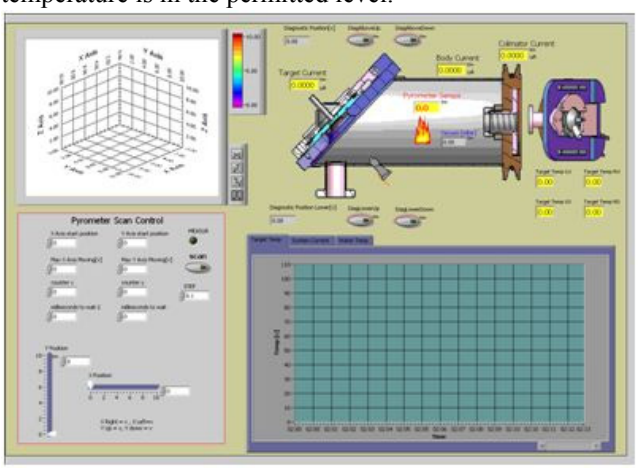


Figure 5: Foil target control system – target parameters.

SUMMARY

The present report gives a brief description of the control systems of the two experiments at the SARAF accelerator temporary proton beam line. Both systems are fully operational, enabling an excellent control of the targets. At the moment, the Foil target experiment is routinely running at SARAF. The LiLIT experiment is routinely in operation, irradiated by an electron gun and will be moved to the proton beam line in early 2012.

REFERENCES

[1] L. Weissman, D. Berkovits, I. Eliyahu, I. Gertz, A. Grin, S. Halfon, G. Lempert, I. Mardor, A. Nagler, A. Perry, J. Rodnizki, K. Dunkel, M. Pekeler, C. Piel, P. vom Stein, A. Bechtold, Proceedings of LINAC 2010, WE102, Tsukuba, Japan (2010).

[2] Mardor I., et al., The SARAF CW 40 MeV proton/deuteron accelerator. Proc. of SRF2009, Berlin, Germany, 2009, 74-80.

[3] Halfon S, et al., High power accelerator-based boron neutron capture with a liquid lithium target and new applications to treatment of infectious diseases. App. Rad. Iso. 67 (2009) S278–S281.

[4] Halfon, S., et al., High-power liquid-lithium target prototype for accelerator-based boron neutron capture therapy. Appl. Radiat. Isotopes (2011), doi:10.1016/j.apradiso.2011.03.016

[5] G. Feinberg et al., A Liquid-Lithium Target project for production of high-intensity quasi-stellar neutrons, 11th Symposium on Nuclei in the Cosmos - NIC XI, Heidelberg, Germany, July 19–23 2010, in Proceedings of Science, http://pos.sissa.it

[6] I. Silverman et al., Production of Palladium-103 from a thin rhodium foil target – Improved cooling concept, Nuclear Instruments and Methods in Physics Research B 261 (2007) 747–750

IFMIF LLRF CONTROL SYSTEM ARCHITECTURE BASED ON EPICS*

Julio Calvo†, Angel Ibarra
Centro de Investigaciones Energéticas Mediomabientales y Tecnológicas, Ciemat, Spain
Miguel Angel Patricio, Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad Carlos III Madrid, Spain
Mark Rivers, Department of Geophysical Sciences and Center for Advanced Radiation Sources
The University of Chicago, USA

Abstract

The IFMIF-EVEDA (International Fusion Materials Irradiation Facility - Engineering Validation and Engineering Design Activity) linear accelerator will be a 9 MeV, 125mA CW (Continuous Wave) deuteron accelerator prototype to validate the technical options of the accelerator design for IFMIF. The primary mission of such facility is to test and verify materials performance when subjected to extensive neutron irradiation of the type encountered in a fusion reactor to prepare for the design, construction, licensing and safe operation of a fusion DEMO (Fusion Demonstration Reactor). The RF (Radio Frequency) power system of IFMIF-EVEDA consists of 18 RF chains working at 175MHz with three amplification stages each. The LLRF (Low-Level Radio Frequency) controls the amplitude and phase of the signal to be synchronized with the beam and it also controls the resonance frequency of the cavities. The system is based on a commercial cPCI (Compact Peripheral Component Interconnect) FPGA (Field Programmable Gate Array) board provided by Lyrtech and controlled by a Windows Host PC. For this purpose, it is mandatory to communicate the cPCI FPGA Board with an EPICS Channel Access [1], building an IOC (Input Output Controller). A new software architecture to design a device support, using AsynPortDriver class and CSS as a GUI (Graphical User Interface), is presented.

INTRODUCTION

The RF System is defined as the equipment necessary to convert the high-voltage AC (alternating current) primary power to suitably conditioned RF power for input to the IFMIF-EVEDA accelerator cavities [2]. The quality of the RF delivered to the accelerator cavities is controlled to within ± 1 degree in phase and to within $\pm 1\%$ in amplitude, using a low-level RF-drive modulated control system. Each RF Module Local Control System (RF Module-LCS) is a device that will monitor and control all physical parameters within the RF Chains located in the same RF Module. Each RF module comprises 2 RF Chains, so the 18 RF Chains will be monitor and controlled by 9 RF Module-LCS connected via Ethernet to the Central Control System (CCS) [3], this Local Control System scheme is shown in Fig. 1. Furthermore, The primary role of the Low Level Radio Frequency system (LLRF) is to control the amplitude and the phase of each cavity (fast regulation) and to control the tuning of each cavity to keep its resonant frequency close to the accelerator operating frequency. For doing so, the LLRF will generate the RF signals (RF drives) for the amplifiers feeding the cavities, depending on their voltage and forward power. The IFMIF-EVEDA LLRF system has to work under CW mode operation and it has also to support pulse mode operation (during the commissioning and tuning of the prototype accelerator) [3].

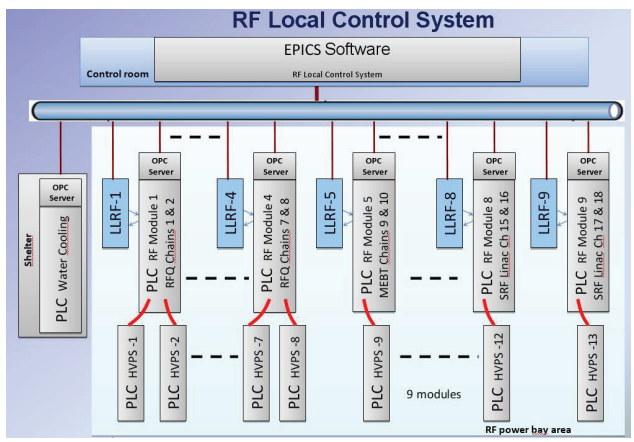


Figure 1: LLRF local control system scheme.

For controlling and operating this LLRF System, IFMIF-EVEDA decided to chose EPICS some years ago. EPICS is a set of Open Source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments such as a particle accelerators, telescopes and other large scientific experiments. Nowadays, EPICS is widely used in big accelerators like Diamond Light Source and huge experiments like ITER. EPICS provides a number of tools for creating and operating a control system. This minimizes the need for custom coding and helps ensure uniform operator interfaces [4]. These features make EPICS the most appropriate architecture for IFMIF-EVEDA, hence, the main purpose of this paper is to present the LLRF control system based on EPICS.

* This work is funded by Ministerio de Ciencia e Innovación del Gobierno de España, under the Project No. AIC10-A-000441.
† julio.calvo@ciemat.es

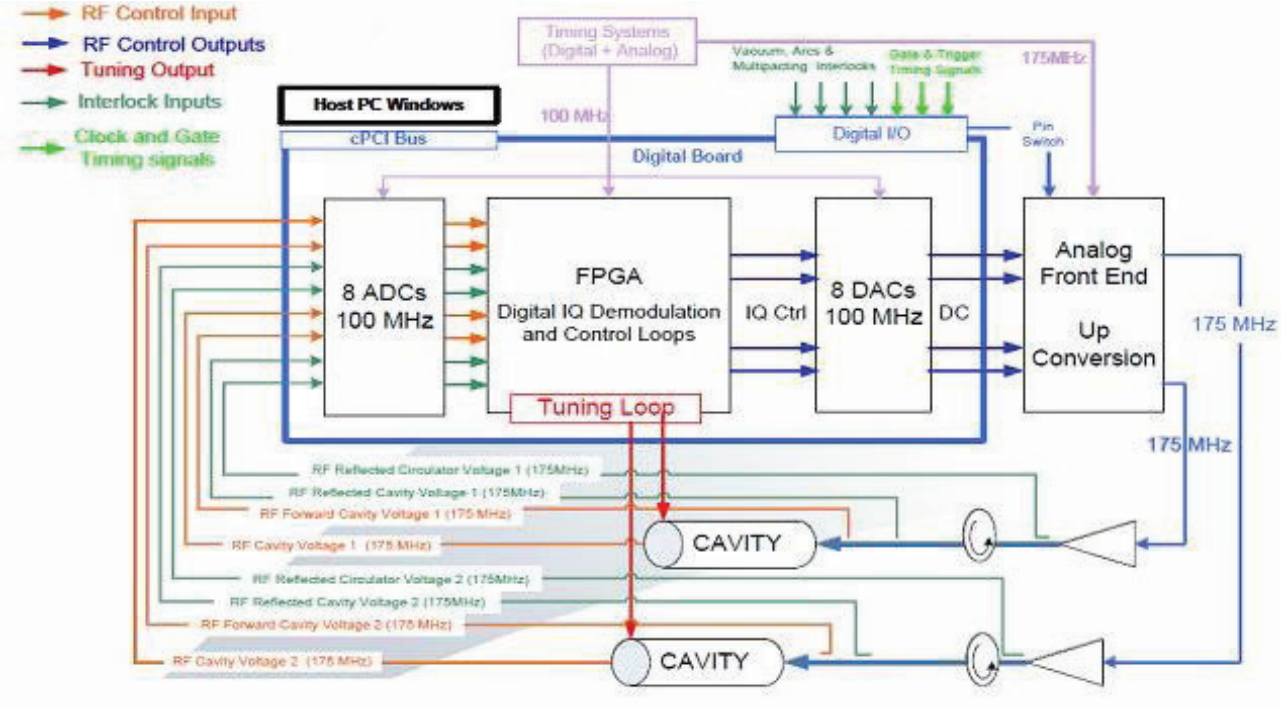


Figure 2: LLRF System general overview.

SYSTEM ARCHITECTURE

Hardware Architecture

The LLRF System is composed by three main subsystems: a digital board with fast FPGA, the analog front end and a local timing system.

- Digital Board: The digital board contains one Virtex-4 FPGA, 8 ADCs and 8 DACs with 14 bits resolution and capable to work up to 105MHz. It is a commercial board with cPCI format provided by Lyrtech and it will be controlled by a Windows CPU. This board will acquire different kind of signals: RF control inputs (cavity voltage and forward cavity power), RF interlock inputs (cavity reflected power and circulator reflected power), digital interlocks (arcs, vacuum and multi-pacting) and timing signals (gate and pulse signals). It will also provide the control outputs of the LLRF: Direct Current (DC) control outputs for signals. It will also provide the control outputs of the LLRF: DC controls amplitude, phase, tuning and interlocks outputs of the cavities. The LLRF general system overview is shown in Fig. 2.
- The front end: It is in charge if up-converting the DC control outputs from the digital board into RF. For doing so, the LLRF employs a quadrature IQ modulator.
- The local timing system: It consists of a PLL board with a 100MHz VCXO (CDC-7005-EVM from Texas Instruments). This board provide 100MHz TTL signal to clock the digital board. This signal will be phase locked with an external 10MHz signal provided by the general timing system.

Sotfware Architecture

The presented architecture is developed over EPICS Base 3.14.11, using Visual Studio 2008 C++ Express Edition, Asyn4-13-1, Application Programming Interface (API) for Lyrtech boards and Control System Studio (CSS).

This solution consists of the following EPICS components:

- IOC (Input/Output Controller): The IOC is any platform that can support EPICS run-time components, including the database, and its access routines, device drivers, record types for various input and output and scanning and monitoring functionality [5].
- Database: The EPICS database [6] is a basic element in an IOC. The database is composed by numerous instances of records. A Record is a collection of Processes Variables (PVs) which is implemented on an IOC where a Process Variable (PV) is a piece of data which can be accessed by its unique name. Often the PV name consists of a record name and the name of a record field, connected by a dot [7]. In the LLRF control system, the records have an associated hardware, Lyrtech boards.
- Device Support: We could define it as an interface between records and hardware. A device support routine has knowledge of the record definition. It also knows

how to talk to the hardware directly or how to call a driver which interfaces to the hardware. Thus, device support routines are the interface between hardware specific fields in a database record and device drivers or the hardware itself [8].

- AsynDriver: It is a general purpose facility for interfacing device specific code to low level communication drivers. A primary target for asynDriver is EPICS IOC device support but it is independent of EPICS [9].
- LAN: Local Area Network. This is the computer based network which allows the communication between IOCs and Operator Interfaces (OPIs). EPICS provides a software component, Channel Access, which provides network transparent communication between a Channel Access client and an arbitrary number of Channel Access servers [5].
- OPI: Control System Studio (CSS) is a combined effort of several parties, including DESY (Hamburg, Germany) and SNS (Oak Ridge, TN). It provides a collection of control system tools in a common environment, based on Eclipse [10].

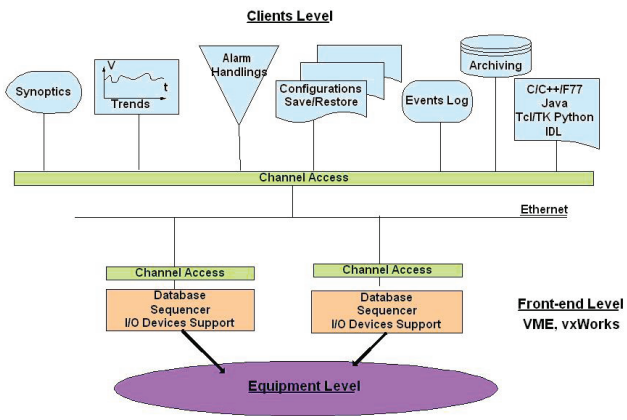


Figure 3: EPICS five layers structure.

This architecture is module based and it follows the EPICS five layer model, this architecture is shown in Fig. 3. In client level we have used CSS as software that allows PVs to be accessed and modified from the network, using the second layer, the Channel Access, that is the communication protocol used by EPICS to transfer information through the network. The next layer is the record support one, where the IOC is settled; here, we have the software which implements PVs for the use with Channel Access. The fourth Layer, device driver layer, it is the heart of the architecture and permits the communication between records in the DataBase and the device. The fifth level corresponds with the instrumentation level, FPGA boards. The proposed architecture is shown in Fig. 4. Due to the Common Software Platform of IFMIF-EVEDA, where all the computers must include the same version of Linux, Vxworks core and EPICS, but cPCI FPGA Board is controlled by a Windows Host PC, carrying out independence of the hardware from the operating system was one of the main challenges of the presented architecture.

USER INTERFACE

The graph user interface, Figure 4, has been designed using CSS. Initially, the proposed GUI package for IFMIF-EVEDA was Extensible Display Manager (EDM). Despite the initial effort that is necessary to implement a clean and robust design, we consider that we made the right decision. We have tried to design a friendly and operative GUI. The operator and the machine engineer can easily tune the system, control the parameters and program Lyrtech boards through the set of designed pages but, following the EPICS philosophy, the whole logic involved have not been developed in the client level, CSS, but on the IOC, then everyone can reach it.

SUMMARY AND FUTURE PLAN

We have developed a Device Support for IFMIF-LLRF Control System. This new software architecture is running properly in a Windows CPU connected in the same chassis of the Lyrtech cPCI board and solves the operating system dependency. This new architecture allows both client and server run in different machines. This solution will be used to control the LLRF of two plants in the final accelerator prototype which will be built in Rokkasho, Japan, on the 2013. Thanks to CSS and EPICS Channel Access, process variables can be seen in any computer of the IFMIF-EVEDA Central Control System. The EPICS based characteristics of the system makes it useful because of its modularity and it can be easily upgradeable and modifiable, the choice of EPICS as a control toolset was very important to achieve this success.

The further work line moves on the testing of the system within the overall prototype accelerator. Else, adding more elements to the GUI and creating an auxiliary database in order to store the historical values of the diagnostics signals is necessary. When these signals are out of certain margins, some warning messages should appear next to the relevant diagnostic signal, these warning messages will be defined later on.

REFERENCES

[1] J.C. Yoon, J.W. Lee, K.M. Ha, J.H. Kim, J.M. Kim and J. Choie. *"EPICS Based Control System for the KOMAC RF System"*. Proceedings of EPAC 2004, Lucerne, Switzerland. Pohang Accelerator Laboratory, POSTECH, Pohang 790-784, Korea, 2004.

[2] International Energy Agency. *"IFMIF Comprehensive Design Report, January 2004"*.

[3] I. Kirpitchev, P. Méndez, M. Weber, A. Ibarra, M.A. Falagan, M. Desmons, A. Mosnier. *"RF Power System for the IFMIF-EVEDA Prototype Accelerator"*. Proceedings of

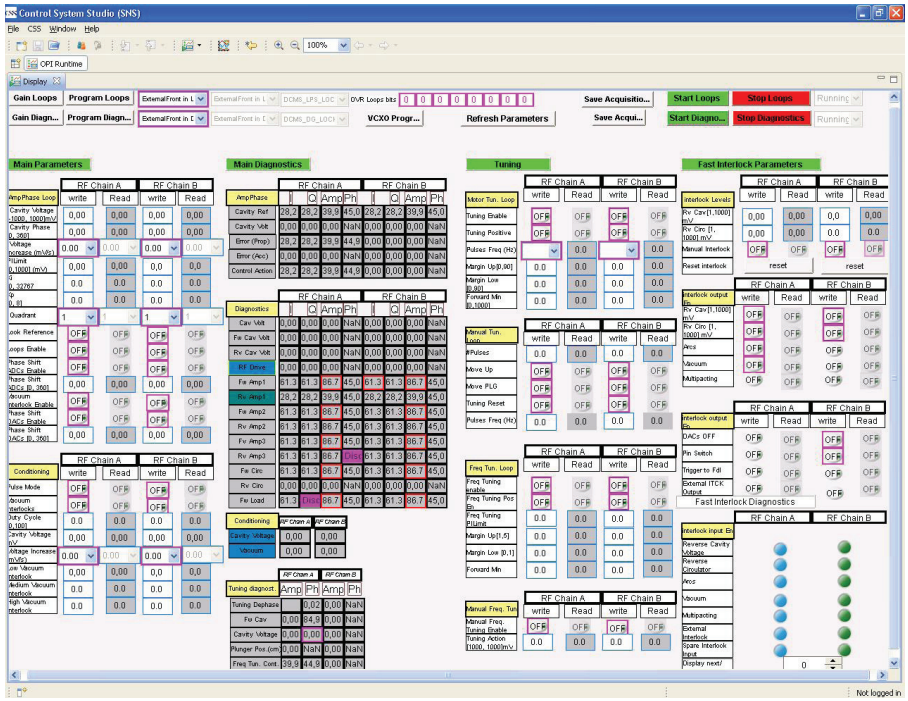


Figure 4: OPI CSS.

EPAC 2004, Lucerne, Switzerland. CIEMAT, Avda. Complutense 22, 28040 Madrid, Spain and CEA, IRFU, F-91191 Gif-sur-Yvette, France, 2008.

[4] EPICS: http://www.aps.anl.gov/epics/

[5] EPICS IOC: http://www.aps.anl.gov/epics/base/R3-14/12-docs/AppDevGuide/node4.html

[6] Stanford Database: http://www.slac.stanford.edu/comp/unix/package/epics/training/documents

[7] EPICS Base: http://epics.web.psi.ch/style/training/handouts/e_basehandouts/node2.htm/

[8] Device Support: http://www.aps.anl.gov/epics/docs/USPAS2010/Lectures

[9] Asyn: http://www.aps.anl.gov/epics/modules/soft/asyn

[10] DESY: http://css.desy.de/

LANSCE CONTROL SYSTEM FRONT-END AND INFRASTRUCTURE HARDWARE UPGRADES

Martin Pieck†, Dolores Baros, Chris Hatch, Pilar S. Marroquin, Peter D. Olivas,
Fred E. Shelley Jr., David S. Warren, and William W. Winton
Los Alamos National Laboratory, NM 87544, USA.

Abstract

The Los Alamos Neutron Science Center (LANSCE) linear accelerator drives user facilities for isotope production, proton radiography, ultra-cold neutrons, weapons neutron research and various sciences using neutron scattering. The LANSCE Control System which is in part more than 40 years old provides control and data monitoring for most devices in the linac and for some of its associated experimental-area beam lines.

In Fiscal Year 2011, the control system went through an upgrade process that affected different areas of the LANSCE Control System. We improved our network infrastructure and we converted part of our front-end control system hardware to Allen Bradley ControlsLogix 5000 and National Instruments Compact RIO programmable automation controller (PAC). In this paper, we will discuss what we have done, what we have learned about upgrading the existing control system and how this will affect our future plans.

BACKGROUND

LANSCE is the proposed site for Los Alamos National Laboratory's (LANL) future signature facility Matter-Radiation Interactions in Extremes (MaRIE). This experimental facility will be the first in a proposed new generation of scientific facilities for the materials community to meet 21st century national security and energy security challenges [1].

Currently different efforts are under way to prepare the facility for its future mission. One effort is based on the LANSCE Linac-Risk Mitigation (LL-RM) project that focuses on risk mitigation of programmatic equipment — components and equipment directly related to the acceleration of the particle beam. Other efforts are focused on parts of the facility that are not directly related to the acceleration of the particle beam.

Despite these different efforts that are mostly driven by different funding sources, there is an overarching strategy for the instrumentation and controls system:

- Replacement of obsolete Controls Hardware
- Installation of new Timing System (in 2013)
- Improvement of Controls Infrastructure

The scope of the work ranges from integration of commercial-off-the-shelf components to complete substitution of obsolete and non-maintainable systems and equipment. For the most part, the projects are self-performed by LANL's Accelerator Operations and Technology - Instrumentation and Controls (AOT-IC) group, which provides maintenance and support of the injector, accelerator, beam transport, and target control system (CS) at the LANSCE User Facility (LUF).

†pieck@lanl.gov

PROJECT EXECUTION CONSTRAINTS

To execute any scope of work, while maintaining 3000h/year beam operation requires the integration of the upgrade projects into the (LUF) operation schedule [2].

The LUF operates cyclically. It typically operates for several months a year to meet commitments to programmatic customers. Run cycles are interspersed with intra-cycle maintenance breaks of a few days to a week for the purpose of recycling the H- ion source and conducting emergent or time-sensitive maintenance. Run cycles are followed by months-long outage; during which scheduled and more extensive maintenance is conducted.

The LANSCE Control System (LCS) which is based on the Experimental Physics and Industrial Control System (EPICS) is critical to operations. Even during intra-cycle maintenance breaks the CS is often needed. Any major upgrade or modification needs to be addressed during the months-long outages. Traditionally this annual outage starts in January and ends in April. During the four month outage in 2011 AOT-IC executed several major projects:

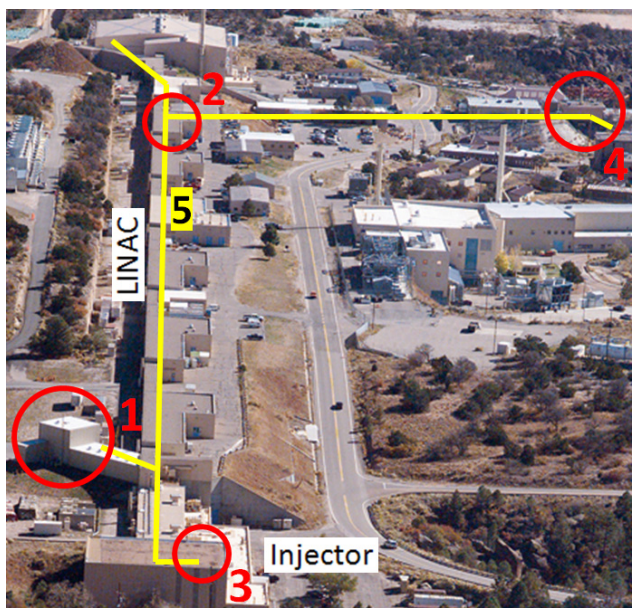


Figure 1: LANSCE User Facility.

1. Isotope Production Facility: Proprietary Control System changed over to an Allen Bradley based Control System
2. LINAC – Sector H: One module of the original "RICE" (Remote Instrumentation and Control Equipment) system was changed over to a National

Instruments cRIO based Control System at the end of the 800MeV Linac
3. PSR – Mechanical Equipment Building: Two CAMAC crates were changed over to a National Instrument cRIO based Control System.
4. H- Injector: CAMAC changed over to an Allen Bradley based Control System
5. Network: Installation of a Fiber Optics back bone communication infrastructure that connects the major areas of the LANSCE Accelerator Facility

These projects will be discussed in turn blow.

ISOTOPE PRODUCTION FACILITY

The Isotope Production Facility (IPF) which produces medical isotopes is part of the LUF. The original IPF Target CS was a stand alone system utilizing non-LCS standardized controls software and hardware components. Due to the fact that the CS was designed, installed and maintained by a non LANL, out of state contractor any maintenance or repair event was time consuming and costly. As a result, IPF management decided to replace the proprietary IPF CS with LCS standardized technology.

The stringent IPF production schedule, as well as the pre and post production requirements, limited the CS upgrade work to 2 1/2 weeks. This included time to take the old CS hardware out, to install the new one, and to test and commission the CS system. In order to accomplish such a daunting task the controls team had to prepare and optimize their approach.

In a process which started 3 months before the actual upgrade work, the old CS was examined during the inter-cycle maintenance breaks. In combination with old CS requirements documentation the system was re-engineered and a suite of process control charts that reflected the operational behavior were developed. These process control charts went through a rigorous review process where operators and expert users verified and validated the new design documentation with requirements and their experience of the old CS. Furthermore, a full CS prototype was developed and tested prior to installation.

For this application an Allen Bradley (AB) ControlLogix 5000 PLC was chosen since the previous system was a PLC as well, though from a different brand. An "EtherIP" driver/device support module interfaces to the AB ControlLogix 5000 PLCs via Ethernet to EPICS IOCs under EPICS R3.13 or R3.14. The software driver (EtherIP) is available through the EPICS collaboration [3].

To accommodate the required Input/Ouput (I/O) points we installed two PLC chassis in the Master-Slave configuration. This saved the project one controller module on the slave side. The Master chassis hosts the AB 1756-L61 ControlLogix 2MB 5561 Processor with an onboard flash memory card that holds the ladder logic program for cases when the internal memory could lose its process application. That same chassis holds all Analog I/O modules of types Spectrum Controls 1756sc-IF8u and AB 1756-OF8. The slave hosts all digital modules of type AB 1756-IB32 and AB 1756-OW16I. The AB's 1756-ENBT ControlLogix EtherNet/IP Module serves as interface between the AB control chassis as well as interface between the EPICS IOC and the AB Master P

The installation and commissioning were successful. However, despite many design reviews, a prototype, and our best effort, some of the requirements surfaced just after the system was installed.

LINAC – SECTOR H

The CS for the 201.25 MHz and 805 MHz linac is based on RICE (Remote Instrumentation and Control Equipment) which was installed in the early 1970's when the facility was built. With more than thirty year old technology come many problems. Many of the discrete components in RICE electronics can no longer be purchased, supplies of spares are dwindling and plastic connectors are becoming brittle. Calibration capabilities were not built-in, raising questions about year-to-year comparability of settings. Lack of any vendor support means all maintenance and repair must be done in-house. Non-standard electronics means all in-house maintenance people must receive extensive RICE-specific training on technology that was installed 40 years ago. The current architecture and implementation is shown below [4].

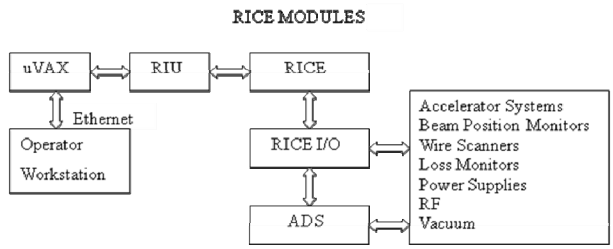


Figure 2: Block Diagram of the RICE System.

RICE controls can be divided into two main categories.

- Industrial Controls: Consists of command and readback of two-state devices, along set-point generation and analog readback of devices.
- Timed-and-Flavored Data: Consists of data taken with a specific relationship to the beam-type (flavor) and timing.

As part of the LL-RM project, a design has been developed to replace the Industrial Controls of the RICE system. The major component of this system are Programmable Automated Controllers (PACs) built by National Instruments. The controller is called CompactRIO (cRIO) and is a reconfigurable control and acquisition system. – An architecture already supported by EPICS [5].

The controller is a cRIO–9024 a Real-Time Controller: 800MHz, 512MB DRAM, and 4GB of Storage. The removable cRIO controller sits in an NI cRIO-9118 chassis with additional 8-slots, Virtex-5 LX110 reconfigurable chassis. The most noticeable advantage is

its high performance and reliability, high performance FPGA, and hot swappable modules. For our application the cRIO is hosted in a commercial BiRIO rack mount chassis which interfaces to commercial BiRIO interface boards that in turn provides interfaces to the device field wiring. This configuration promises in many cases a simple chassis swap during 24-7, on-call operations support while the diagnostic and repair would occur in the electronics laboratory setting.

Figure 3: cRIO in BIRO chassis.

For our analog signals we use the NI cRIO-9205, 16-Bit Analog Input module. The NI cRIO-9474 Digital Output Module is used to create 5 V, 50us wide pulses for stepper motors. For our digital signals we use the NI cRIO-9425 Input Module and the NI cRIO-9477 Digital Output module. A block diagram of the new system is shown below:

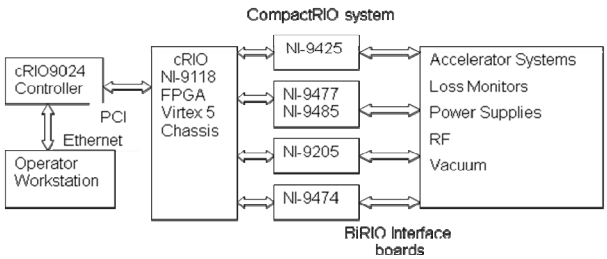


Figure 4: Block Diagram of the cRIO System.

While the controls replacement work was a success the new control wiring needs improvement. Since we replaced only the Industrial Controls I/O of the Rice system the rewiring of the intermixed signals created an eye sour. For the future, any controls rewiring will be done for both the time flavored and industrial control data at the same time. This will minimize any rewiring of the time flavored system when it will be installed in 2013.

H- INJECTOR

The former H- injector CS was based on 1980's Computer Automated Measurement And Control, (CAMAC) technology. It is a modular data handling system and its primary application is data acquisition. Individual crates are controlled by slave or intelligent controllers. The controllers are tied together with a parallel Branch Highway that ends in a Branch Driver. The Branch Driver is interfaced directly to a data acquisition computer; in our case an EPICS VME IOC.

With technology that was installed in the 1980's comes a host of problems. Often the documentation for individual modules is unavailable, spare inventory is limited, many modules are not available anymore and repair of modules is difficult or impossible due to lack of suitable spare parts.

Modern Ethernet enabled data acquisition technology is capable of replacing the CAMAC functionality. Based on our positive experience with AB we did choose the rugged Control Logix5000 PLC for this job in which 80kV arc downs are part of the daily routine.

For this project we did install 2 AB PLC systems, one for the 670kV another for the 80kV section of the Injector. Despite the fact that these PLC were only about 6 feet apart from each other, due to the voltage difference and the required process control communication, a fiber optics network link between the PLCs had to be installed.

The information exchange was implemented through AB ControlLogix producer and consumer tags. Unfortunately, AB's PLC consumer tags do not recognize if the communication to the producer has failed. It just keeps the last know state. Since the communication link between the two PLCs is critical for Run Permit and interlocks, we installed a watch-dog timer that alerts the system and operator if the information link is broken.

The AB PLC is basically the same that we used for the IPF system upgrade with one exception. We introduced an AMCI 5274 High Speed Inspection Module that operates as a stand-alone module with continuous reporting of status information to the AB 1756 system. It can take analog samples at known points; defined by position, time, or digital input, and compare the measured values to a programmed profile and determine the pass/fail of the inspection. We used the AMCI module as a triggered ADC to read the pulsed signals that were previously read by a sample and hold chassis to measure two current monitor signals as well as the current signals of High Voltage and Arc Power Supply.

Unfortunately, operations of the AMCI modules within the arc prone environment caused the module to lock up. Attempts to improve the situation within the high voltage H- Injector Dome environment failed. Even with good vendor costumer support we were unable to find a solution to reset the module without having to reboot the whole PLC.

Since this was not practical solution to this problem we installed the PLC AMCI module in Injector Control Room and connected the signal to the AMCI through fiber optics cables. This solution has proven to be reliable even though it was not our preferred solution.

MECHANICAL EQUIPMENT BLDG

The Mechanical Equipment Building (MEB) hosts a variety of beam line control devices that help transport and diagnose the beam to the different experimental areas (1L and WNR). Part of the in 1980's installed CAMAC system was replaced by a NI cRIO system of the same type as used for the LINAC Sector-H. Leveraging the development and prototyping effort that was done for the LL-RM, project the controls team was able to replace 2 out of 4 CAMAC crates at minimal incremental cost.

For the maintenance outage in 2012 we do expect to replace the remaining two CAMAC systems freeing up more spares for other CAMAC systems still in use.

NETWORK UPGRADE

The communication network upgrade is the first LL-RM scope element that has been delivered. The new network is the foundation for most other system upgrades that will be installed over the next 5 years.

Since the early 1980's, our old LCS network has grown in a piece-by-piece fashion from no network at all to network in many areas. At its core it has a 100Megabit switch and some of the edge switches have only 10Megabit. In many cases we have unmanaged switches in place which makes maintenance, troubleshooting, and performance monitoring inefficient or impossible. Furthermore, the historically grown structure has little to do with modern network topologies. In fact it consists of a multi star configuration where one edge switch functions as a "core switch" for many other new edge switches. Many of these switches were connected via single or multi-mode fiber or CAT5 twisted pair cables.

Our new installed fiber optics network has 10 gigabit (Gb) capabilities at the core switches and 1 Gb at the edge switches. Near the Central Control Room (CCR) we installed five Nortel-5530 core switches. Each switch has 12x1Gb fiber optics ports, 24 x 1Gb twisted pair, and 2x10Gb fiber ports (Note: After installation we learned that 12 of the 1Gb fiber ports and 12 of the 1Gb twisted pair ports are using overlapping resources. The use of one fiber optics port reduces the availability of one twisted pair port.

The new communication network is a true star configuration. In total, 451 fiber optic strands all originating near the CCR were installed. Each of our 42 primary drop off locations has anywhere from 12 to 24 fiber strands. In total 12,800 feet of single mode fiber was installed throughout the accelerator facilities. At each primary drop off location either a Nortel-5510 or 5520 edge switch was installed. Both have 24x1Gigabit twisted pair ports and 2 or 4x1GBit fiber ports respectively. (Note: The Nortel-5520 switch has Power over Ethernet). Near these switches we ran between 12 to 48 CAT6A twisted pair patch cords to secondary drop off locations. These patch cables can be connected back to the edge switch as needed. Under this regime 21,000 feet CAT6A twisted pair cable was installed for this project.

Currently we are in the process of developing a centralized configuration and monitoring schema. The related work including the transition from the old to the new network will start during our next planned maintenance outage in 2012.

CONCLUSION

Commercial off-the-shelf hardware can replace purpose-built hardware with a customized I/O system that interfaces directly with modern standardized control software. RICE controls for two accelerator modules and, a few CAMAC crates in one beam line and injector have been successfully replaced with 8 cRIO Industrial I/O and Allen Bradley systems with more to follow.

REFERENCES

[1] R. W. Garnett and M. S. Gulley, "Matter-Radiation Interactions in Extremes," LINAC'10, Tsukuba, Japan September 2010.

[2] Martin Pieck et. al. Systems Engineering Aspects to Installation of the Phased Multi-Year LANSCE-R Project, 2009 ICALEPCS, Kobe, Japan, (2009), www.JACoW.org

[3] Information on the Allen Bradley EitherIP Driver: http://ics-web.sns.ornl.gov/kasemir/etherip/

[4] S.C. Schaller, "A LAMPF Controls Retrospective: The Good, the Bad, and 'It Seemed Like a Good Idea at the Time'," 1993 Int. Conf. on Accel. and Large Experimental Physics Control Systems, Nucl. Instr. and Meth. A 352 (1994) 516-520.

[5] E. Björklund, A. Veeramani and T. Debelle, "Using EPICS Enabled Industrial Hardware Control Systems," ICALEPCS'09, Kobe, October 2009, WEP078, p.555(2009); www.JACoW.org

PROGRESS IN THE CONVERSION OF THE IN-HOUSE DEVELOPED CONTROL SYSTEM TO EPICS AND RELATED TECHNOLOGIES AT ITHEMBA LABS*

Ivan Kohler, Mogamad Amien Crombie, Cheslin Ellis, Mike Hogan, Hendrik Mostert, Maria Mvungi, Camelia Oliva, John V. Pilcher, Nieldane Stodart, iThemba LABS, Faure, South Africa

Abstract

This paper highlights challenges associated with the upgrading of the iThemba LABS control system. Issues include maintaining an ageing control system which is based on a LAN of PCs running OS/2, using in-house developed C-code, hardware interfacing consisting of elderly CAMAC and locally manufactured SABUS[1] modules. The developments around integrating the local hardware into EPICS, running both systems in parallel during the transition period, and the inclusion of other environments like Labview are discussed. It is concluded that it was a good decision to base the underlying intercommunications on channel access and to move the majority of process variables over to EPICS given that it is at least an international standard, less dependant on a handful of local developers, and enjoys the support from a very active world community.

INTRODUCTION

The cyclotrons of the iThemba Laboratory for Accelerator-Based Sciences have been in operation for approximately 25 years, and provide particle beams for basic and applied research, hadron radiotherapy and the production of radioactive isotopes.

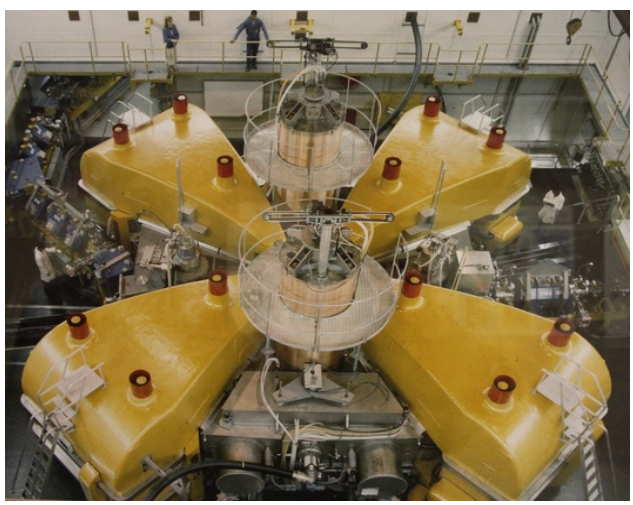


Figure 1: The Separated Sector Cyclotron commissioned 1987.

The laboratory is also responsible for two 6MV Van de Graaff accelerators (a single and a tandem).

*Work supported by National Research Foundation of South Africa

The Control System in Brief

To meet the operational requirements of the facility, the current control system has evolved into a number of subsystems largely based on an Ethernet LAN of low cost PCs running various operating systems with customised application code.

There are +/- 3000 devices that need to be controlled resulting in about 30000 process variables. A distributed database was developed in-house with different parts of the database being housed in node PCs near to the equipment and which communicate to console PCs in various control areas manned by operators. There are approximately 100 OS/2 based PCs 10 if them providing MMI[2] to operators and about 90 acting as local controllers providing I/O to the various devices, each maintaining tables of variables making up parts of the distributed database. The operator interface is presented by an in-house developed console program that displays lists of variables.

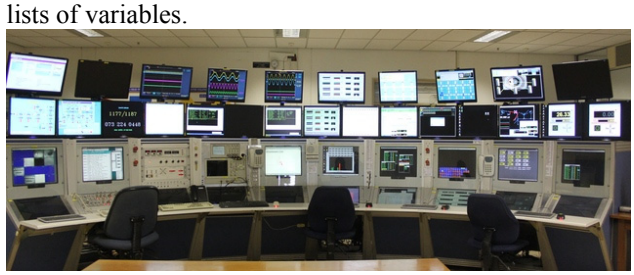


Figure 2: The renovated Control Desk.

Problems with the Current System

The OS/2 operating system is no longer supported by IBM. A generic called eComStation[3] has been used in the interim, though it cannot compete with the support and advances made in systems like Linux and Labview. The CAMAC hardware is difficult to find and expensive to maintain. Software maintainability has become an issue as the developers who wrote the original code retire or move on.

Constraints

It was decided to implement the new system within a number of constraints. A local area network of low cost standard personal computers would be used. An open source Linux operating system and open source EPICS tool-kit would be used wherever practical. The extensive infrastructure linking devices to crates, both CAMAC and SABUS via an 8-bit parallel differential bus would remain on account of the large amount of re-cableing that would have to be done if this was changed. The CAMAC crates

would slowly be replaced by SABUS crates as the new control system developed.

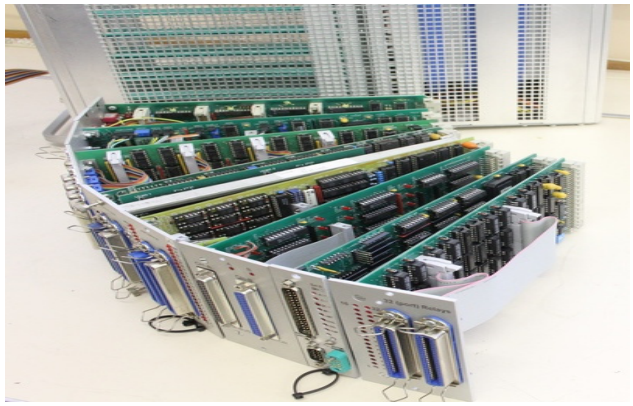


Figure 3: A variety of SABUS cards and a crate, most connectors compatible with CAMAC modules.

Challenges

The first challenge was to develop an understanding of EPICS and then to write an interface to the locally developed PC ISA and PCI master differential bus driver cards through the ASYN layer. The next was upgrading the SABUS interface cards, designing them to operate with EPICS and currently available ICs. Given the availability of limited manpower, and the continuous operation of the facility, a major challenge was to provide for both old and new control systems to operate simultaneously. A further challenge was finding out every aspect of the old system.

PROGRESS SO FAR

Currently the range of devices controllable by the new PC-based EPICS IOCs using SABUS cards, include power supplies (also RS232 serial), stepping motors, pneumatic actuators, all aspects of vacuum system control, slits, scanners and a terminal stabilizing unit for a van de Graaff accelerator. A .NET-based Windows program has been developed to capture harp wire currents and display beam profiles. Labview is used in the logging and monitoring of the charge integrating system of the target stations and in both water and Helium target cooling systems. A software bridge operating in both directions between old and new systems has been developed so that channel access is provided to the old OS/2 variables and the older OS/2 console programs can interact with the newer EPICS based process variables.

EPICS IOCs

A Linux device driver for PC-based differential bus cards was developed followed by a C++ intermediate program which makes calls through the EPICS ASYN layer to this driver. EPICS record based template files were developed for each of the SABUS IO cards, which include a power supply controller, a stepping motor controller, an actuator driver card, a 32 way relay card, an 80 bit opto isolated input card, an 8 channel 16 bit ADC card and an equivalent of a CAMAC IGOR[4] card. DAC functions are handled by the power supply controller card.

A script file (ioc-install) was written so that an IOC can be assembled with little effort. To build an IOC from scratch a user:

- installs Ubuntu 10.04, chosen as it has LTS[5] status for the next three years,
- runs the script file, which will install EPICS base, ASYN, MEDM and configure an IOC based on makebaseapp.pl to include record template files and make everything,
- edits st.cmd and db/substitution files to customise the IOC thus setting up for the number and types of process variables controlled by the IOC.

Start up script files are also generated by the ioc-install script which allows the IOC to start executing on PC power up.

Before After

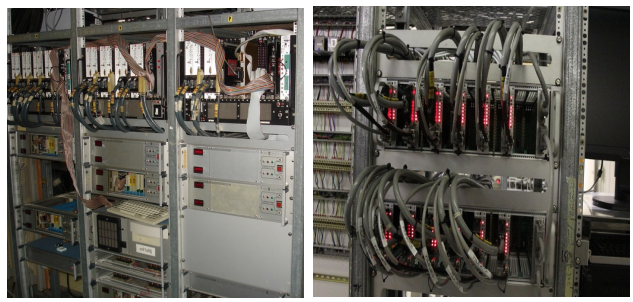


Figure 4: 3 CAMAC crates plus driver boxes reduced to 2 SABUS crates and 12 modules.

The Vacuum Control System

The EPICS vacuum control system is meant to be a direct replacement of the old OS/2 software control system, but with the flexibility to adapt in an easier way to changing I/O hardware, electronic interface equipment and beam-line configurations in the future.

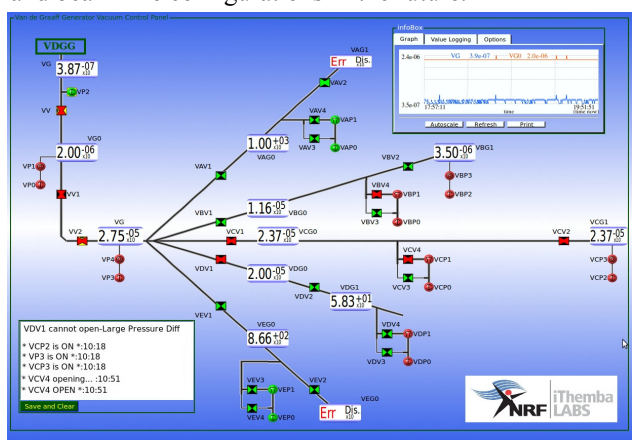


Figure 5: Display of the Van de Graaff Vacuum panel, similar designs will be implemented on the cyclotrons and related beam lines.

The old vacuum system had to remain in operation and having been proven to be conceptually sound, the new vacuum system was modelled on the old.

New EPICS-friendly 32-bit Relay-Output cards were designed. These units provide 32 individually addressed bits whereas the previous modules were byte addressed.

The way the new system is set up is meant to facilitate quick and easy switching between the old and the new vacuum systems during the installation and initial live testing phase.

For the GUI, customised EPICS-capable plug-in widgets/GUI components were created. Using Qt Designer, these GUI-objects can be dragged and dropped into a display and those with EPICS functionality can have their Process Variable names entered. Customised Signals and Slots easily link and transfer information between customised components and standard ones.

With the decision to move to EPICS, the control logic and electronics interfacing is abstracted away from the GUI which affords developers the opportunity to keep up with GUI trends.

The Harp System

The new Harp System consists of the following subsystems: a soft-IOC holding "soft channel" waveform records of harp data, a Harp server application that updates the waveform records and any number of EPICS clients accessing data from the soft-IOC.

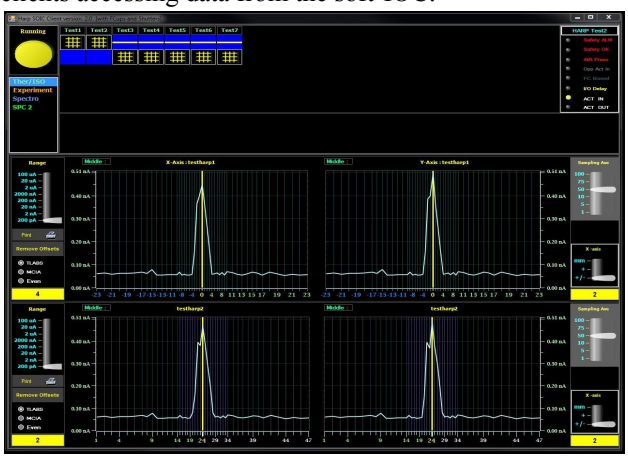


Figure 6: A display for 2 harps and actuators on beam-lines.

The Harp server is the interface between the harp hardware and the soft-IOC. The server monitors actuator process variables and as soon as any client moves a harp into the beam, the server connects to the harp hardware and starts reading ADC values (2 X 48 channels) into the soft-IOC. An EPICS client reads and displays (Fig 6) the ADC values. It also control other pneumatic devices like faraday cups, neutron shutters and attenuators.

The Harp server has been developed on Microsoft.NET Framework 3.5 (using C#) and National Instruments Measurement Studio in VS 2010.

The harp hardware is controlled by a Rabbit[6] (8-bit) micro-controller (programmed with Dynamic-C) that sends TCP/IP packets from the harp's hardware, to the Harp server. PSI's DOTNET library (C#), is used for updating the soft-IOC.

Labview Systems

Labview is used at iThemba as a rapid development system where post digital signal processing and interfacing to off the shelf hardware are a requirement. The ease of use of the Labview interface to existing EPICS IOCs has been a great help. Labview was chosen for the upgrade of the isotope production target stations control system.

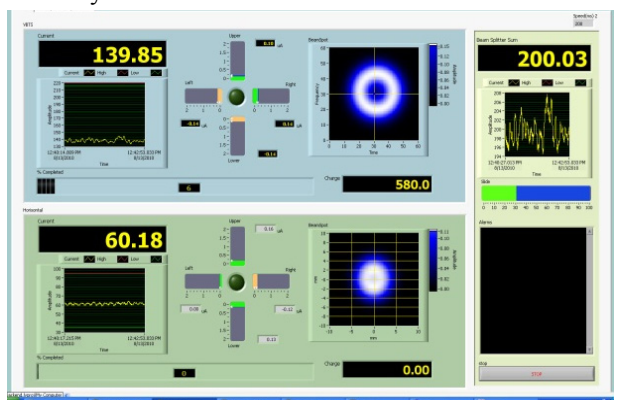


Figure 7: Display of the operator interface to the vertical and horizontal target stations with 200uA split beam.

Currently Labview is used for the logging and monitoring of the charge integrating systems of the horizontal and the high intensity vertical target stations(Fig 7). The software is configured as a main hardware client which stores data to a logging server and Labview shared variable clients for local and remote-control and viewing of target current, collimator currents, beam sweep parameters and collected charge.

The water target cooling system and the Helium target cooling system have been ported from old MSDOS Pascal systems to National Instruments compact RIO FPGA using Labview as the programming interface.

The Scanner System

The program scanner provides a display for National Electronics beam profile monitors. Up to 16 scanners may be connected to National Instruments cards managed by an EPICS IOC. The IOC must be configured with a custom EPICS ASYN driver, developed at iThemba LABS for this purpose.

The program interfaces to the IOC through Channel Access. Clients can be run on different desktops for the same set of scanners. More than one copy of the program can be run on the same desktop to display different sets of scanners, these sets need not be located on the same IOC, since access is only through the process variable names. This program uses Qt Open Source Edition version 4.5.0. by Nokia

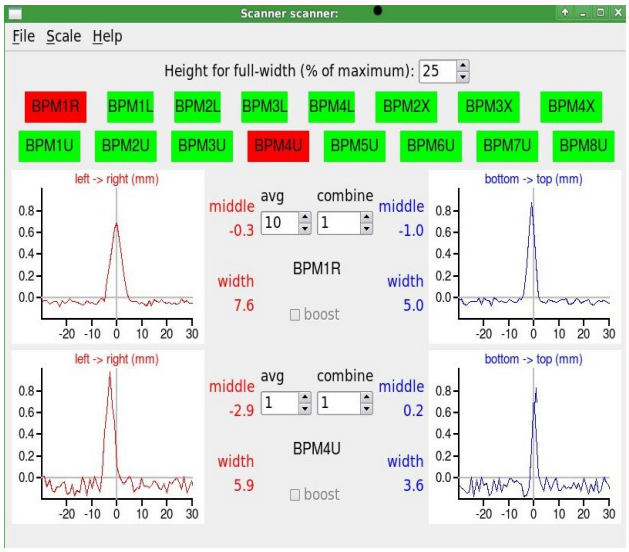


Figure 8: A display showing X and Y profiles of 2 scanners.

Bridging Between Old and New Systems

The old control system [7] is still in use while the new EPICS system is being slowly deployed. It was found necessary for new EPICS programs to communicate with the old system. This was achieved by writing software that acts as a bridge between the systems. The bridging software consists of two programs. The first program uses EPICS channel access server software to get requests for process variables. The program then communicates with the second program, using TCP/IP, that runs on the old control system. The old control system variables appear as EPICS process variables that new programs can handle.

A Switching Program Between Therapy and Isotope Production Beamlines

An IOC is used to control switching of the beam between Therapy and Isotope lines. This is used for beam sharing between radio therapy treatment and radioisotope production. The algorithm, written with EPICS records, goes through a list of checks when setting up the individual lines and aborts the process if any of these fail.

An Energy Table Machine Set Up Program

A Java based program has been developed to allow sections of the accelerators and beam-lines to be speedily and efficiently set up for different beam energies. The program uses a MYSQL relational database. Energy settings are easily managed and after process variables are written to, they are checked for proximity to their set values and alerts given where necessary.

EPICS Clients Currently Being Used

The Archive System is based on the newer Control System Studio (CSS). An Archive Engine takes mainly Vacuum Pump PV data samples via Channel Access and places them in a MySQL Relational Database (RDB). Besides the CSS Data Browser, a JSP-based web report of data is also used to view archived data and allows such data to be plotted and/or exported.

IRMIS, Gateway and Alarm Handler programs will only become fully operational when more systems have been upgraded to EPICS.

The Terminal Potential Stabilizing Unit

A terminal potential stabilizing system including electronic interfacing hardware, EPICS process variables and MEDM control screen was developed for the single ended 6MV van de Graaff accelerator which has greatly lead to improved stability of the beam.

CONCLUSION

The split of implementation is about 90% non-commercial (EPICS and in-house cards) to 10% commercial (Labview, .NET and NI[8] cards). Where there is a large number of I/O cards required (for example about 150 dual power supply controllers) it is cost effective to go in-house. To date about a third of the process variables have been transferred to EPICS on Ubuntu Linux, using off-the-shelf personal computers and in-house interface cards. They are proving to be stable. A consequence of a non-commercial solution is that a fair amount of time has had to be allowed for the development of the software drivers for the I/O cards as well as manufacture and debugging of the I/O cards. This has removed the reliance on commercial suppliers though and the flexibility of the local designs has allowed tailoring of interfacing to match any equipment already installed. Once the design phase is complete the cost per system is low. However on the commercial side when it has been determined that smaller quantities of more specialized interface cards are required Labview in conjunction with these cards has proven to be a very convenient tool with excellent graphics, ready-made drivers and a wide range of I/O cards available. Cognisance also has to be taken of the available skills within the development team. Some developers prefer the Microsoft .NET environment and having experience in this technology allows for rapid development of their assignment, an example being the harp and faraday cup system, successfully deployed.

REFERENCES

[1] SABUS - a collaboration between Iskor (PTY) Ltd. and CSIR (Council for Scientific and Industrial reseach) (1980)
[2] MMI – Man Machine Interface
[3] eComStation - Serenity Systems International
[4] IGOR Input Gate Output Register, a CAMAC module
[5] LTS – Long Term Support (3 years) by Canonical
[6] Rabbit Semiconductor, (RCM-3365)
[7] Present Status of the NAC Control System ICALEPCS99 Theron,Hogan,Kohler,Krijt,Schulein, van d Merwe,van Niekerk.
[8] National Instruments, headquarters, Austin, Texas

GSI OPERATION SOFTWARE: MIGRATION FROM OpenVMS TO Linux

Ralf Huhmann, Günther Fröhlich, Susanne Jülicher, V. R.W Schaa, GSI, Darmstadt, Germany

Abstract

The current operation software at GSI, controlling the linac, beam transfer lines, synchrotron and storage ring, has been developed over a period of more than two decades using OpenVMS on Alpha-Workstations. The GSI accelerator facilities will serve as an injector chain for the new FAIR accelerator complex for which a control system is currently developed. To enable reuse and integration of parts of the distributed GSI software system, in particular the linac operation software, within the FAIR control system, the corresponding software components must be migrated to Linux. Interoperability with FAIR controls applications is achieved by adding a generic middleware interface accessible from Java applications. For porting applications to Linux a set of libraries and tools has been developed covering the necessary OpenVMS system functionality. Currently, core applications and services are already ported or rewritten and functionally tested but not in operational usage. This paper presents the current status of the project and concepts for putting the migrated software into operation.

MISSION

Presently, the operation software at GSI runs on a cluster of DEC-Alpha[1] machines. The computers' OS is OpenVMS. The user interface is realized by X-Window and Motif based clients. There are some hardware display and control units, i.e. proprietary interfaced LED and LCD displays, knobs, analog gauges, etc. The software modules for operation of the linear accelerator (LINAC), the transfer lines, and common services are mainly written in Fortran77. The central software components for operation of the synchrotron (SIS) and the storage ring (ESR) are written in Pascal. All programming languages use specific DEC extensions and a lot of OpenVMS specific system calls. The X11 and Motif implementations embed the X11 events into the OpenVMS system's event scheme. We aim to get rid of OpenVMS but to reuse the main parts of the GSI operational software's source code base by porting the software to Linux. Additionally, the migration shall enable interoperability with current or future developments, e.g. in Java on Linux. It showed up that most of the operation applications written in Fortran77 could be ported to Linux with minimal modification effort utilizing the techniques, libraries, and tools described in [1]. Some other Fortran components, especially system oriented services were rewritten in C/C++/Java on Linux. The components for SIS and ESR written in Pascal could not be ported, mostly because no suitable compiler is available which handles the DEC language extensions.

[1] Alpha is today a brand of Hewlett-Packard.

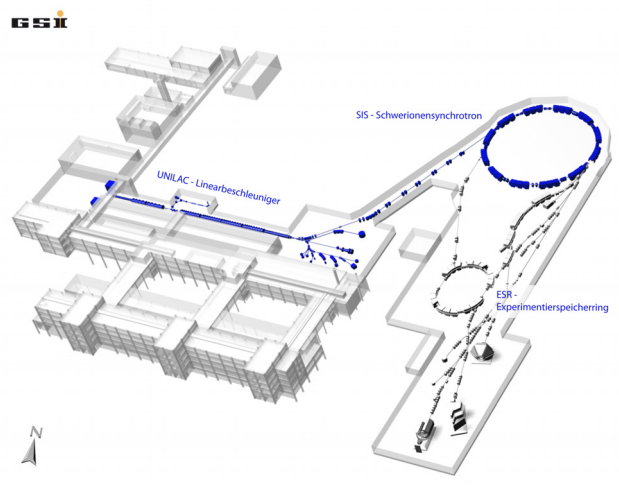


Figure 1: GSI accelerator overview.

ISLANDS

Fortunately, software components for operation of the SIS and ESR on one hand, and the LINAC and transfer lines on the other hand, are only minimally coupled (Fig. 2). Therefore, in step 1 a gateway mechanism was developed to bridge these islands. The SIS and ESR operation software will remain unchanged on OpenVMS and will be excluded from migration in step 1. In a second step those components will completely be replaced by the LSA framework [2] with new Java applications.

New operation applications for the LINAC and transfer lines are developed in Java on Linux and can be bridged with the island of ported Fortran software.

BRIDGES

Fortran@Linux – Fortran@OpenVMS

On OpenVMS, the coupling between SIS/ESR operation software and the LINAC software is realized by a service which sends binary messages from one application to another. It utilizes VMS mailboxes and raw ethernet communication. Actually, for the ported applications on Linux this API was implemented utilizing peer to peer TCP/IP communication and a nameservice (Fig. 4 shows a GUI monitoring the nameservice) to resolve the IP-address and IP-port by the application name. In order to connect from VMS to this framework a proxyservice was developed on

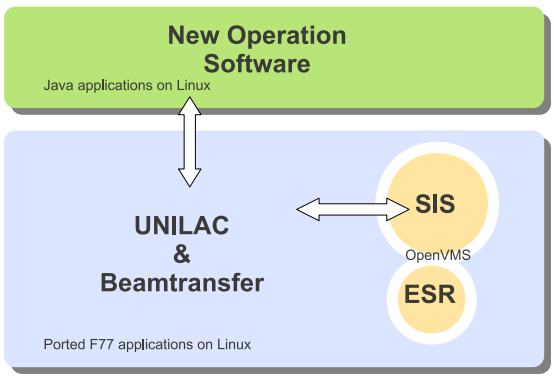


Figure 2: Groups of operation software components in step 1 of migration.

Linux. In Fig. 3 you see as an example two OpenVMS applications `V1` and `V3` which connect via TCP/IP to a service on Linux acting as a proxy for `V1` and `V3` and registering those in the nameservice with the proxy's address and port. Messages from `V1` or `V3` to `L1` or `L2` on Linux are sent to

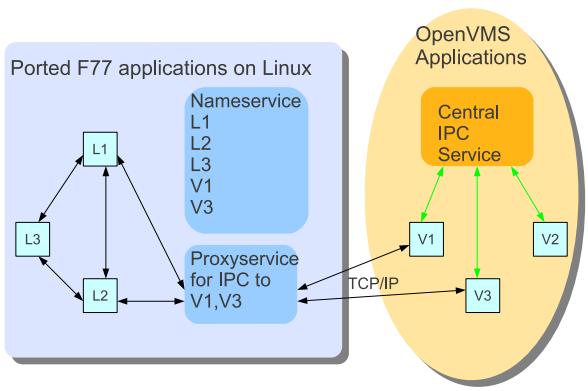


Figure 3: Binary data exchange between Linux and OpenVMS.

the proxy which transmits the packet to its destination application using the name service resolution on Linux. Vice versa, a message from `L1` or `L2` to `V1` or `V3` is received by the proxy since it registered `V1` and `V3` with the proxy's address and port in the nameservice. The proxy forwards the message to OpenVMS via the corresponding TCP/IP socket connection.

Fortran@Linux – Java@Linux

By the *uv*-architecture (Fig. 5) which is described in more detail in [1] an observer pattern is realized to connect Java applications to Fortran applications for data exchange. Fortran applications serve as publisher, Java applications subscribe to data objects. Requests, replies, and notifications are serialized over XML-streams.

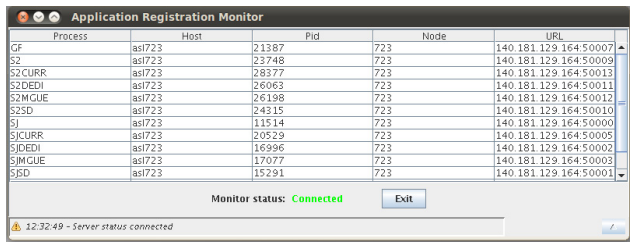


Figure 4: GUI monitor of nameservice.

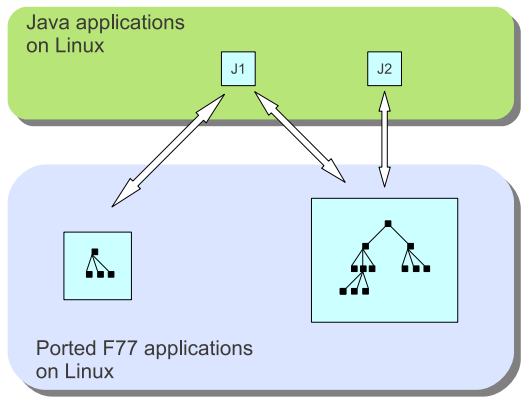


Figure 5: *uv*-architecture for data object exchange between Fortran and Java applications.

Publisher The simple server-API for the Fortran application allows to create and change at runtime a value tree-structure (like adding files and folders to a file-system), to change locally the numerical values of the leaf-nodes, and to put data objects (i.e. arbitrary *uv*-trees) in event-queues. Services like tree transfer of a read result, value changes by clients, sending value change notifications to clients, processing structure changes and subscriptions are hidden in the implementation of the API and are transparent for the application. Value changes initiated by a *uv*-client are notified to the application via callback.

Subscriber The client-API for the Java application is to read the value tree structure or subtree structures. It subscribes to folders for receiving structure changes or to leaf-nodes for receiving numerical value changes or to event-queues for receiving complex data objects. It allows setting values of leaf-nodes of an *uv*-server. Figure 6 shows as an example the value tree of a application seen by a generic value browser utilizing the Java client-API.

CURRENT STATUS

Currently (middle of 2011), the main operation applications for the LINAC and transfer lines are ported to Linux using the VMS system emulation libraries [1]. Some services have been reimplemented, e.g. a central alarm service, a service to launch and monitor applications on selected beam lines, communication modules for dedicated

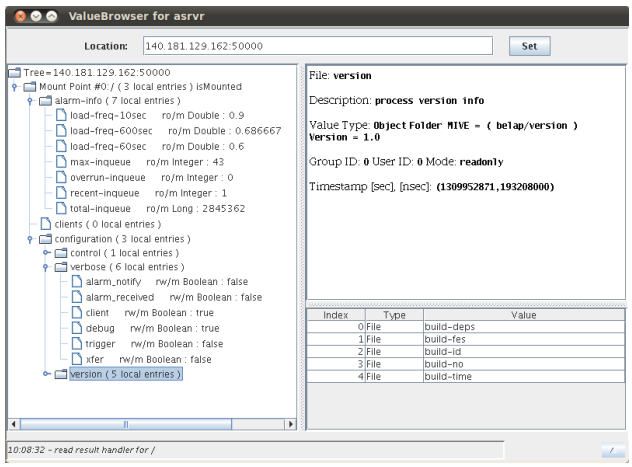


Figure 6: Value tree browser.

hardware control units in the main control room, and a library for accelerator device access. A framework to build the complete software stack from a tagged subversion repository was developed. The above mentioned services and APIs to bridge to remaining VMS software and to new Java applications are implemented and tested.

MIGRATION PATH

Still missing are full integration tests of all operation software components, a production run time environment on Linux, deployment to the production system, performance tests under production conditions, and control room tests without and with beam.

Currently, a test console is set up which will serve as a test site outside the main control room for integration tests and run time environment. In order to switch the operation software to Linux it will be necessary to perform several temporary integration tests in the main control room (Fig. 7), first without beam, later with beam. Hence, to sustain normal operation in the meantime, a procedure has to be defined to enable switching between operation on Linux and back to OpenVMS till the final quality standards are reached. This switching procedure must cover the access to the hardware control units of the consoles, and the interoperation with the left OpenVMS software components. Figure 8 shows the scheduled time scale of the remaining migration path.

Figure 7: Main Control Room at GSI.

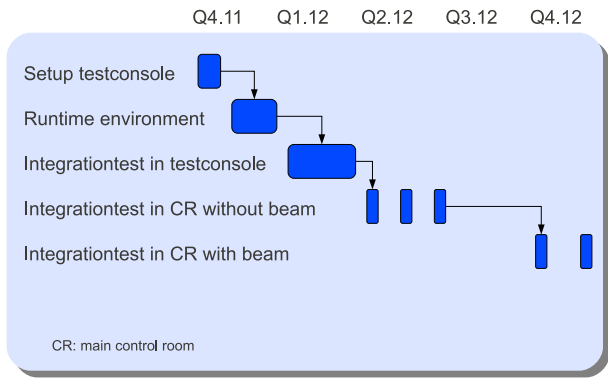


Figure 8: Schedule for migration of operation software.

CONCLUSION

Now, the feasibility studies and most of the implementation, porting, and unit testing has been completed. However, integration and launching in production environment demand further efforts.

REFERENCES

[1] R. Huhmann, G.Fröhlich, S. Jülicher, V.RW Schaa, "GSI Operating Software Migration Openvms to Linux", Proceedings of PCaPAC08, Ljubljana, Slovenia, MOX02, p. 4.

[2] J. Fitzek, R. Mueller, D. Ondreka, GSI, Darmstadt, Germany, "Settings Management within the FAIR Control System Based on the CERN LSA Framework", Proceedings of PCaPAC 2010, Saskatoon, Saskatchewan, Canada, WEPL008, p. 63.

THE CONTROL SYSTEM OF CERN ACCELERATORS VACUUM [CURRENT STATUS & RECENT IMPROVEMENTS]

P. Gomes, F. Antoniotti, S. Blanchard, M. Boccioli, G. Girardot, H. Vestergard;
CERN, Geneva, Switzerland
L. Kopylov, M. Mikheev; IHEP, Protvino, Moscow, Russia

Abstract

The vacuum control system of most of the CERN accelerators is based on Siemens PLCs and on PVSS SCADA. After the transition from the LHC commissioning phase to its regular operation, there has been a number of additions and improvements to the vacuum control system. They were driven by new technical requirements and by feedback from the accelerator operators and vacuum specialists.

New control functions have been implemented in the PLCs; new tools have been developed for the SCADA, while its ergonomics and navigation have been enhanced.

INTRODUCTION

The CERN Accelerator Chain

To minimise the interactions between the accelerating beams and the residual gases, and thus maximise the beam lifetime, the beam pipes of all accelerators must be pumped down to a suitable vacuum level.

The Large Hadron Collider (**LHC**) at CERN is a 27 km proton accelerator-collider, with two separate beam pipes that merge at the interaction points. The static vacuum pressure is of about 10^{-11} mbar; due to beam dynamic effects, it can rise up to 10^{-8} mbar.

In order to achieve high fields for accelerating and guiding the proton beams, superconducting materials are used. Therefore, magnets and RF cavities are operated at temperatures down to 4.5 or 1.9 K; they are fed by a separated cryogenic distribution line (QRL).

For thermal insulation, both the superconducting magnets and the QRL are immersed in vacuum below 10^{-5} mbar, often reaching 10^{-7} mbar. The beam and insulation vacuum systems of the LHC totalise some 109 km.

Feeding the LHC, the Super Proton Synchrotron (**SPS**) and its transfer lines add 16 km of beam pipe vacuum [10^{-7}..10^{-9} mbar]. Going back through the chain, the Proton Synchrotron (**PS**), its Booster (**PSB**) and the Linacs (**L2**, **L3**) add a further 2 km.

Vacuum Gauges

To read pressures from atmospheric down to 1 mbar, simple membrane gauges (**VGM**) are used. Thermal conductivity (or Pirani) gauges (**VGR**) are linear in the range 1..10^{-3} mbar. Cold cathode ionisation (or Penning) gauges (**VGP**) reach further down [10^{-5}..10^{-11} mbar].

To cover a wide pressure range, a pair of Pirani and Penning are assembled together and read by common electronics, becoming a "full-range" gauge (**VGF**).

Hot cathode ionisation (Bayard Alpert) gauges (**VGI**) are used in the range 10^{-7}..10^{-12} mbar.

Vacuum Pumps

Primary pumps (also called “roughing pumps”) are used to evacuate from atmosphere down to 10^{-3} mbar; they are also used as a backing pump for Turbo Molecular pumps (TMP), who are effective in the range 10^{-3}..10^{-10} mbar. Often they are both assembled and controlled together as a Pumping Group. If their usage is temporary, they can be are embarked in a mobile trolley (**VPGM**), together with the powering and control electronics. For long-term pumping (**VPGF**), only the pumps and powering circuitry are left close the accelerator, while all the radiation-sensitive electronics is kept in safe areas.

When needed to reach ultra-high, vacuum sputter-ion pumps (**VPI**) are added [10^{-5}..10^{-11} mbar]; as a bonus, their current consumption is a good measurement of the pressure.

Sublimation pumps (**VPS**) are used in the range 10^{-9}.. 10^{-12} mbar. Cryogenic pumps are used in particular cases.

Vacuum Valves

To isolate a pumping group from the pumped volume (vacuum sector), roughing valves are used (**VVR**). Sector gate valves (**VVS**) provide isolation between vacuum sectors, to prevent leak propagation, or to allow for a mechanical intervention. There are also window valves (**VVW**) and fast shutter valves (**VVF**).

Instrument Count in Accelerators Vacuum

All in all, (Table 1) there are more than 6 000 instruments to be controlled and monitored, distributed along 128 km of vacuum, in the range of 10^{-4}..10^{-11} mbar. The mobile pumping groups are not counted here; nor are the gauges and valves in the fixed pumping groups.

Table 1: Vacuum Equipment in CERN Accelerators

	L2,L3, PSB,PS	SPS	LHC beam	LHC insul.	other facilities	total
length [km]	2	16	59	50	1	128
log (P [mb])	-7..-10	-7..-9	-8..-11	-5..-7	-4..-10	-4..-11
PLC master	5	8	28		3	44
PLC other	0	10	7		0	17
PLC slave	0	0	100		155	255
VGM	0	0	10	231	0	241
VGR	102	113	428	348	61	1052
VGP	122	128	649	364	66	1329
VGI	28	0	167	0	16	211
VGF	0	13	4	0	0	17
VPGF	7	3	14	179	51	254
VPI	370	1429	825	0	69	2693
VPS	48	0	0	0	0	48
VVS	76	87	305	39	13	520
VVF	0	11	0	0	0	11
VVW	0	5	0	0	0	5

VACUUM CONTROLS ARCHITECTURE

The CERN accelerator complex has a history rich of several decades; vacuum controls architecture and equipment have been evolving, with the construction of every new accelerator machine and with the availability of new technologies.

Given the different evolution paths of each machine, managed by different entities, and depending on the funding availability, it was not obvious to always enforce homogeneity of architectures, equipment and naming. It is common to find several versions of the same equipment, from different generations, coexisting in the same machine. In order to increase the efficiency in debugging and repairing and to simplify the stock management, a considerable effort is being put into analysing the existing versions of each equipment and selecting only a couple of them, that would comply with most of the requirements.

Stand-alone Locally Operated Vacuum Controls

In the early times, a central control room near the accelerator concentrated all electromechanical actuators and displays; they were directly cabled to the machine and were not accessible from elsewhere.

That is still the case of some experimental facilities around the PS Complex, where the electromechanical equipment has been replaced by electronic front-ends, but continues to be accessed only locally.

VME & X-Windows / JAVA

In the 90's [1,2], the vacuum controls of the PS Complex and SPS were renovated: for each facility, a VME crate (DSC - Device Stub Controller) collected the data through an RS232/X25 interface; either directly, from industrial controllers equipped with a RS232 port (VGR, VGP, TMP), or through G64 concentrators, for custom-made equipment (VVS, VPI, VPS). The DSC was remotely accessible through Ethernet, on Unix workstations running X-Windows and MOTIF as graphical user interface. The pumping groups, however, were kept running in stand-alone mode.

Until the next wave of renovation, this is still the case of the PS Ring, AD (Antiproton Decelerator) and CTF3 (CLIC Test Facility), although with new graphics provided by Java ("Working Sets").

PLC & PVSS

Since 2000, the SPS, then the LHC (Fig. 1) and part of the Complex PS have been upgraded to a PLC-based architecture [3,4], using the Siemens™ S7-400 series. The human-machine interface is a SCADA (Supervisory Control And Data Acquisition), built with PVSS®.

The application software for both PLC and PVSS started to be custom-made by the vacuum group. With time, it included a growing number of building blocks from the UNICOS framework [5].

Geographically limited accelerators or installations are controlled by a single PLC. Wider machines have one PLC at each underground service area (Complex PS: 5, SPS: 7; LHC: 28). Independently of the number of PLCs used, there is only one PVSS Data-Server (DS) per accelerator complex.

The PLCs and the DS communicate through Ethernet in a protected and restricted "Technical Network"; consoles in the "Office Network" can have access limited to monitoring the evolution of vacuum variables.

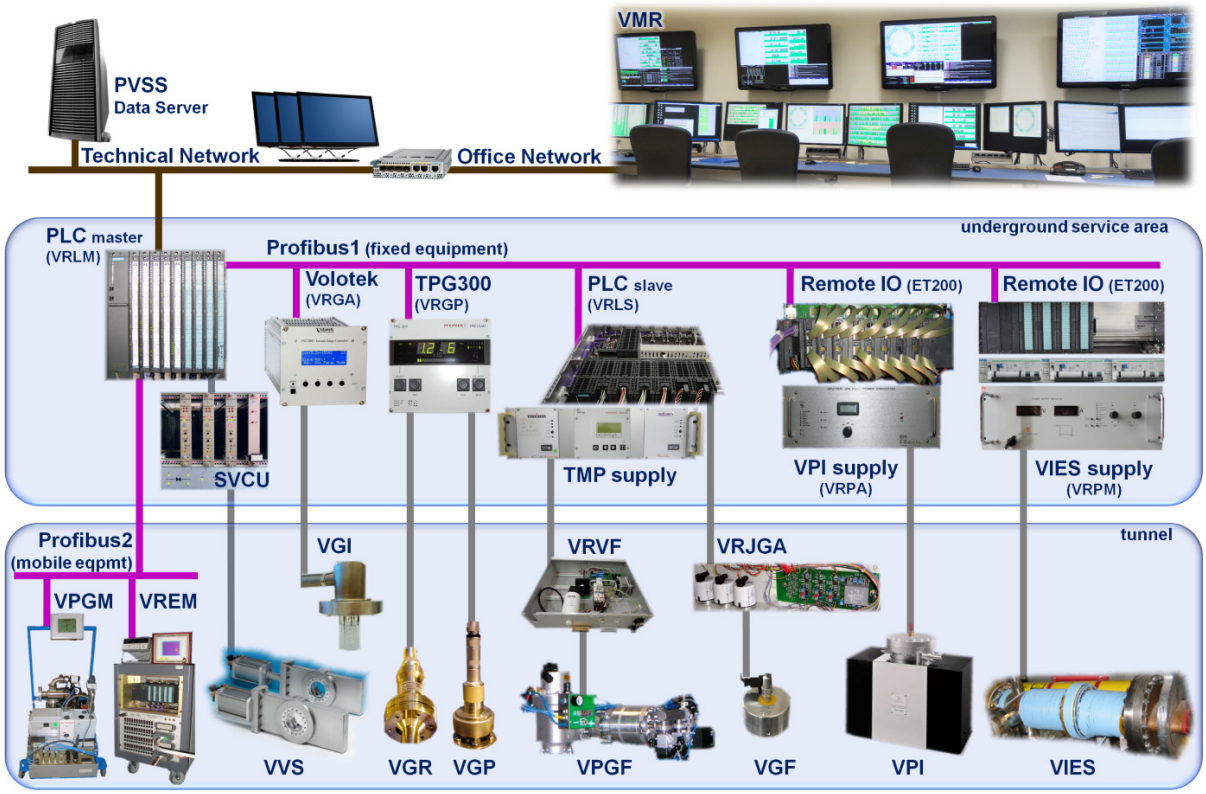


Figure 1: LHC vacuum controls architecture.

Equipment in the Underground Service Areas

A PLC accesses the field equipment (gauges and pumps) through controllers or power supplies; the modern ones are often intelligent, as they have an embarked microprocessor or other programmable device, like an FPGA; when equipped with the corresponding interface, they can communicate with the PLC via Profibus® (a serial communication link for field equipment); this minimises the complexity and price of cabling and also allows for a wider exchange of information and configuration parameters. This is the case of the controllers for VGI (Volotek), for the VGR and VGP (TPG300), and for the power supplies of the recently added "anti-electron-cloud solenoids" (VIES); also, the fixed pumping groups (VPGF) and their TMP controller are managed by a small "Slave" PLC (S7-300), connected to the "Master" PLC by Profibus.

On the other hand, the controllers for the VVS are directly connected to individual IO channels on the PLC; the power supplies for the VPI are connected to remote-IO stations (Siemens-ET200).

While the PVSS-DS reside on centralised surface buildings, the PLCs, controllers and power supplies are kept in underground service areas, away from the accelerator tunnels where radiation could damage them.

Equipment in the Tunnel

In the accelerator areas where the estimated radiation was considered as acceptably low, and where the distance to a service area was quite large, "active" gauges were installed. These consist of a VGF (pair of VGR+VGP) with its front-end electronics nearby. Individual VGR and VGP ("passive" gauges) that are closer to the radiation-protected areas can be directly accessed by a remote TPG.

A dedicated dynamically-configured Profibus network connects the Master PLC to the "mobile" equipment; although sensitive to radiation, they only come to the tunnel when the machine is shut-down. Examples are the mobile pumping groups (VPGM), managed by another "Slave" PLC (S7-200), and the mobile bake-out stations (**VREM**) handled by an S7-300.

Interlocks, Alarms & Warnings

To restrict potential leaks, if several pressure readings (VGP, VPI) in the same vacuum sector rise above a given threshold, an interlock is sent to the VVS controller; the neighbouring sector valves will then close, confining the vacuum sector. To limit its sensitivity to noise or spurious variations of pressure, this interlock is a voting combination of the relay outputs of TPG and VPI_supply: if a majority of the readings is OK, no interlock is produced. However, if at least one relay is not-OK while the neighbouring valves had been closed for some reason, the valves cannot be opened again until all relays are OK. The details of the voting logic are machine-dependent.

As the automatic closure of the VVS would interfere with the beam circulation, when a pressure interlock is going to close the valves, an interlock is also sent to the beam control system (BIC), in order to previously dump the beam. Conversely, the "beam ON" information from the BIC prevents the manual closure of the VVS.

Hardware interlocks are also sent to the control systems of cryogenics, RF, MKI and MKD, in case of degradation of the respective isolation vacuum.

Devices like the VPI, VGP and VGI may be degraded or even damaged if operated at too high pressure; their controllers have internal protection interlocks to power them off in case their pressure reading rises above a given threshold; for the VGP, the interlock can also be derived from the associated VGR within the same TPG.

Less critical situations, but nevertheless important to other control systems, are sent from PVSS to the LHC Alarms Service (**LASER**), to be handled by the accelerator operators.

Vacuum issues demanding the attention or intervention of an expert generate also an alarm in PVSS, which will send a mail or SMS to the corresponding list of experts.

The PVSS application provides a wide set of visual warnings, in terms of colours and widget animations, to draw the attention of the operators in case of unusual conditions.

Communication with Other Systems

Process status and values are published by PVSS, to the other control systems, through the middleware interfaces **CMW** or **DIP**. Also, PVSS can get data from CMW and DIP. Alarms for the accelerator operators are published to LASER. The corresponding configuration comes from **CCDB** (the Controls Configuration database).

The PVSS own local archives are limited in size, and therefore in time-depth, depending on local disk capacity; therefore, the historical data is periodically sent to a central repository - the **LOGGING** database.

Databases & Software Generation

A set of ORACLE databases contain the information necessary to automatically generate the equipment description for the PLCs and for PVSS.

The **Master DB** contains the parameters common to all the machines. This includes the definition of **equipment types** according to the physical characteristics. In addition, equipment is also classified by **control types**, with a set of attributes required to interface the equipment with the control system. Often, equipment of the same type are controlled in the same way.

The **Machine specific DBs** (LHC DB, SPS DB, CPS DB) contain information about each machine, such as the definition of all vacuum sectors, and the individual attributes of each installed equipment. The information about the geographical distribution of all vacuum equipment is imported from the Layout and Survey databases. A user-friendly Java application (**DB_editor**) allows the manual entering or modification of any individual attribute.

The **DB_export_tool** combines the information from the above data bases and produces the configuration files for both PLC (DataBlocks) and PVSS (DataPoints, CMW and DIP servers, LHC Logging, LASER, etc.).

NEW FEATURES

PLC

The control of the pumps and valves in a pumping group is the result from a simple logic combination of several status inputs, including pressure levels, together with some timers. After a power failure, some of the status inputs are lost and the group has to be restarted manually. This is particularly annoying if a thunderstorm causes a power cut in a large region of an accelerator.

A palliative solution was implemented, that includes the cyclic memorisation of the status inputs; when the function detects that the power of the group was off and back again, it tries to automatically restart it, taking into account the memorised status inputs.

In parallel, a completely new approach is under development, using the concept of sequential state machine. Here, the outputs of the function depend only on the current state of the machine, not on the input values. The state is the memorised resultant of the sequence of previous states and inputs. At any time it will be easy to restart from the memorised state.

In order to reduce the electron-cloud effects in the beam vacuum, several solenoids have been installed around critical parts of the LHC. A new function was developed to control this new type of equipment and allow the operators to manually pilot their current from PVSS.

To fight unexpected leaks in the insulation vacuum, it became necessary to quickly install a pumping group with the electronics kept in radiation-free zones. The automatic recognition and integration of slave PLCs into the mobile Profibus has been extended for this purpose.

PVSS

In order to improve the ergonomics and efficiency of diagnostics, several improvements were made on the existing PVSS tools. The users can now define and save the full details of their own email/SMS notifications, one by one or globally in a table. Also the historical trends can now be configured and saved by the users. The query for device list has been improved and merged with the query for state/value history, within a given time frame.

To increase security, the PVSS application now starts with user "monitor", who has no rights to manipulate anything. This way, the user ID is not anymore inherited from the current Operating System session. Only when the operator actually needs to perform any action, he will log into PVSS to gain his access rights; after a period of inactivity he will be automatically logged off. This way, we minimise the time a console can remain unattended with a full-rights account.

The values of the thresholds for the interlocks produced by the TPGs can be accessed and modified from PVSS. In times of accelerator study and development, those interlock levels may be changed by the experts. In order to keep track of these modifications, a new tool was developed to automatically back-up of the parameters of all the TPG, every day.

In order to improve the diagnosis of the vacuum systems and the organisation of preventive actions, new panels were created and others were enriched (Fig. 2). The equipment of the PS-Ring and AD, although still on the previous generation of control architecture, is now available for monitoring on PVSS, using CMW data.

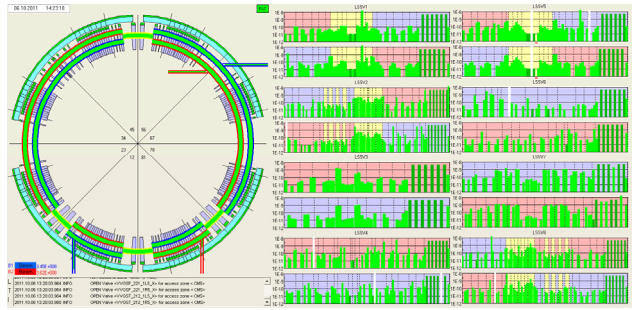


Figure 2: LHC synoptics and pressure bargraphs.

A web server was created, showing PVSS panels that summarise the values and status of the vacuum systems. The function that lists all interlocked valves has been upgraded and exports a list to the web server.

A monitoring room (**VMR**) was set, with 4 stations directly on the technical network and 4 wall-screens running the web summary pages. Soon, all PLCs and DS will be available on **DIAMON**, a tool for remote surveillance and diagnostic of all CERN controls infrastructure.

The PVSS graphical interface for windows clients has been upgraded from the old ActiveX to QT. This finally allows to smoothly move the PVSS servers from Windows to Linux, and also to have client applications on Linux machines.

CONCLUSIONS

New functions have been implemented in the PLC and PVSS. The ergonomics and configurability of the PVSS application have been enhanced.

There is still a long way ahead, regarding the homogenisation of equipment and controls across machines, the convergence towards UNICOS, and the tools for tracking of events, interventions and repairs.

REFERENCES

[1] R. Gavaggio et al., "The New Vacuum Control System of the CERN PS Complex", ICALEPCS95, Chicago, October 1995.

[2] L. Kopylov et al., "A data driven graphical user interface for vacuum control applications", ICALEPCS95, Chicago, October 1995.

[3] R. Gavaggio et al., "Development of the vacuum control system for the LHC", ICALEPCS05, Geneva, October 2005.

[4] P. Strubin "Vacuum controls and interlocks", CERN Acelerator School - Vacuum in Accelerators, Platja d'Aro, Spain, 2006; CERN-2007-003, pp. 369-388.

[5] E. Blanco et al., "UNICOS CPC v6: evolution", ICALEPCS11, Grenoble, October 2011.

NEW TIMING SYSTEM DEVELOPMENT AT SNS*

Doug Curry, Xihui Chen, Richard Dickson, Steven Hartman, David H. Thompson,
ORNL, Oak Ridge, TN, USA
Joze Dedic, Cosylab, Ljubljana, Slovenia

Abstract

The timing system at the Spallation Neutron Source (SNS) has recently been updated to support the long range production and availability goals of the facility. A redesign of the hardware and software provided us with an opportunity to significantly reduce the complexity of the system as a whole and consolidate the functionality of multiple cards into single units eliminating almost half of our operating components in the field. It also presented a prime opportunity to integrate new system level diagnostics previously unavailable for experts and operations. These new tools provide us with a clear image of the health of our distribution links and enhance our ability to quickly identify and isolate errors.

INTRODUCTION

The Timing System at SNS is based on the dual timing system architecture used at the Relativistic Heavy Ion Collider (RHIC) at Brookhaven Nation Laboratory (BNL) [1]. The first system is a beam synchronous system that distributes and correlates specific tasks that are used in the production of the beam pulse. It produces a bi-phase mark encoded signal to embed the ring RF clock with the event scheduling data to provide remote platforms with a reference in which local PLLs can lock to and derive references that are synchronized to the ring frequency. This system was altered slightly to meet the specific requirements at SNS to synchronize the 60 Hz operation of the LINAC and neutron choppers to the accumulator ring [2]. The second system or Real Time Data Link (RTDL) serves as a general purpose information distribution network. It globally broadcasts information across a unidirectional link about each beam pulse. The delivery of this data only has to occur within a specified range of time and is not required to be correlated to the ring frequency and is therefore asynchronous to the ring RF source.

The hardware delivered by BNL can no longer be reproduced as many of the components have reached their end of life cycle. This makes it impossible to support the existing architecture for any extended period of time. A hardware redesign is required in order for us to effectively support the long term availability goals of the facility. This has provided us with an ideal opportunity to simplify our system architecture and focus on decreasing our maintenance overhead by integrating the functionality of multiple components into a single units and making system diagnostic capabilities available at each node.

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy

TIMING MASTER HARDWARE

The Timing Master Hardware achieved the largest gains in reducing individual component design complexity by integrating both of the RHIC timing systems into a single VME card [3]. The previous platform requires multiple sets of hardware components that span across 2 VME crates using a VME bridge [4].

Event Timing

The RHIC beam synchronous system that generates the machine cycle sequence events across the Event Link (EL) requires 4 Trigger modules that produce reference pulses to 4 Event Input modules that are connected to a single Event Link Encoder module using a special adaptor cable that bridges the user defined pins of the P2 connector. The functionality of these 9 cards now resides within the SNS Timing Master Hardware.

The 4 Event Input modules produce all of the fixed or hard events that are used in the production of the beam pulse. The number of fixed events is limited to 64. This restriction is imposed by the Input modules which are only capable of generating 8 events per node. Special care must be taken when setting up this system to prevent Event Jostle. This occurs when 2 or more of these modules try to fire an event at the same time. The generation of soft or floating events must be held off by the IOC during the period of time when beam events are active to ensure they do not cause additional scheduling conflicts while the fixed events are in the process of being broadcast on the EL.

The SNS design utilizes a local memory to store the hard events. It has a depth of 16k address locations. There are only 255 usable events available in the SNS Timing System; therefore, the majority of the memory locations are unused. The Timing Master hardware maintains a counter that represents the current turn count within the machine cycle production period. A turn is defined as the amount of time it takes for the beam to travel around the SNS ring 1 time. There are approximately just over 17k turns in a given SNS machine cycle. This counter is used as an address pointer to the hard event memory. When a non-zero value is returned from a particular address location soft event requests are queued and held off until a conflict is no longer present. The machine specific event numbers are stored in memory locations that correspond to a given turn in the machine cycle when they are scheduled to be broadcast on the event link. Additional events are allowed to pass through the system at anytime there is not a machine specific event scheduled for production. The

possibility of Event Jostle is completely eliminated in this architecture as there is only a single source of the hard events, and the processing of soft events are postponed and queued by the hardware when the address pointer returns a non-zero value. This also allows the IOC to schedule the soft events at anytime within the machine cycle as it is no longer restricted to non-beam production periods. Mapping the fixed events into memory also allows the new system to define any number of the 255 events as either hard or soft events. It is no longer restricted by the Event Input module limitations.

Data Link Timing

The RHIC RTDL hardware requires a single RTDL Encoder and multiple RTDL Input Modules. Each Input module is only capable of producing 8 frames. The encoder and input modules also require a special adaptor cable to bridge the user defined pins of the P2 connector. The protocol defined for the RDTL restricts the unique number of data packets it is capable of transmitting to 256 frames. The SNS system has been configured to broadcast 128 frames of data. This architecture would require a third VME crate if the SNS requirements where updated by adding additional frames above the current 128 that are presently defined and would require installing a 3rd VME crate as a total of 32 Input modules are needed to populate 256 frames.

The SNS Timing Master design eliminated the need for the encoder and input modules. This was handled in a similar fashion to the EL by mapping these frames into a separate local memory bank; however, unlike event driven timing system, this memory only needs to be large enough to contain all 256 frames. The IOC updates values in the RTDL between machine cycles. The RTDL section is processed by the hardware when it observes and RTDL valid event from the IOC. The hardware then processes the entire RTDL table and only queues the frames that have been updated by the IOC. After the table has been processed, the hardware clears the frame table to prevent stale data from being rebroadcast across the link.

TIMING RECEIVER HARDWARE

The most significant achievement in simplifying the overall system architecture was accomplished in the redesign of the Timing Receiver Hardware [5]. The RHIC Trigger Module [6] and the Utility Module [7] were integrated into a single VME card. This has the highest impact on reducing the total number of supported operating components in the field as a large portion of the subsystems utilize both sets of this hardware platforms.

Trigger Module

The primary function of the Trigger Module is to support the SNS subsystems that are responsible for the production of the beam pulse by producing machine cycle specific reference pulses that are synchronous to the ring RF clock. These cards contain local PLLs that derive reference clocks from the bi-phase mark encoded event link generated by the Timing Master hardware at 32x the ring frequency.

These cards contain 8 front panel trigger reference outputs that can be configured independently to produce reference pulses to any of the 255 predefined SNS events. These pulses can be delayed from the start of an event using 1/64 increments of the ring RF frequency for systems that require alignment with the head or tail of the beam pulse within the accumulator ring.

Utility Module

The Utility Module provides the IOC with cycle to cycle information about each beam pulse that is extracted from the RTDL. The module will interrupt the IOC after receiving an RTDL valid event from the Timing Master on the EL. This notifies the processor that the table has been updated and is ready for any user applications that require this information. It also serves a platform for generating additional interrupts to the IOC when specific events have been observed on the EL or an error was observed on the event or real time data links.

System Diagnostics

One of the major challenges with the previous Timing System is determining the existence of and isolating transmission errors. The modules will notify the processor that an error exists; however, it provides very few details about the error. Additional information must be acquired by adding additional VME cards to the configuration.

The new Timing Receiver Hardware incorporates detailed diagnostics for both of the timing links within the hardware design. It utilizes 2 different memory banks that store the observed events from the EL and the frames from the RTDL for each machine cycle. The processor is interrupted only if this feature has been enabled in the hardware. This allows us to field diagnostic tools with every module without running additional application overhead if the diagnostic tool is not needed.

TIMING DISTRIBUTION NETWORK

The SNS Timing Distribution Network also suffers from component availability issues. This hardware redesign has also been modified from the original equipment that was delivered to SNS. This includes changes that actually complicated the design slightly; however, the functionality remains the same. Special care was taken to ensure the health monitoring equipment does not impact the operation of the distribution network should it experience a failure.

The overall distribution architecture was revisited. The previous distribution method performs multiple media conversions prior to reaching its final destination. This adds additional unnecessary jitter to the distributed signal that is received by the end components in the field making it difficult to perform high resolution beam reference measurements. The new design retains the

information on fiber links until it reaches the end point before converting it into an electrical signal.

The new hardware takes advantage of using hot swappable SFP modules and includes an additional new set of diagnostics that allows us to perform health checks of the links at the fan-out modules. This gives us an opportunity to perform preventative maintenance checks during scheduled maintenance periods.

CONCLUSION

By simplifying the system architecture, the SNS Timing System is easier to manage and has significantly fewer active components that are likely to fail. The addition of diagnostics at each node allows us to isolate and identify failures remotely and more rapidly than before. These changes also allow us to incorporate preventative maintenance routines into our scheduled maintenance periods and provide us with the opportunity to address failures before they impact accelerator operations.

Allowing the hardware to manage more of the overhead task allows a number of the IOC responsibilities to be reduced and simplifies the design of the software as well. A fair number of the tasks have been reduced to simply updating a handful of registers in the timing hardware.

The integration of our hardware components has allowed us to remove 8 different types of VME cards from our inventory and combine their functionality into just 2 cards. The Timing Master VME crate previously spanned across two 11U 21 slot VME crates. It now occupies a single 4U 9 slot crate. We have also managed to eliminate approximately 150 active VME cards from our operating environment.

REFERENCES

[1] B. Oerter, "Accelerator Timing at the Relativistic Heavy Ion Collider," ICALEPCS 1999

[2] J R. Nelson, B. Oerter, T. Shea and C. Sibley, "SNS Timing system," ICALEPCS 2001

[3] Doug Curry, "Design Criteria for the Timing Master VME Card," Internal Document 105020000-DC0005, R00, April 2011

[4] Eric Bjorklund, Dave Thompson, "SNS Timing System Technical Description," Internal Document 109020200-TD0001, R00, April 2003

[5] Jeffrey Patterson, "Design Criteria for the Timing Receiver VME Card," Internal Document 105020000-DC0003, R02, February 2011

[6] H. Hartmann, "Specification for the V124S Trigger Module," Internal Document 109020200-TS0009, R00, November 2000

[7] P. Stein, "SNS Utility Module Functional Description," Internal Document 109020200-TS0004, R00, November 2000

HIGH INTENSITY PROTON ACCELERATOR CONTROLS NETWORK UPGRADE

R. Krempaska, A. Bertrand, F. Lendzian, H. Lutz,
Paul Scherrer Institute, 5232 Villigen PSI, Switzerland

Abstract

The High Intensity Proton Accelerator (HIPA) control system network is spread through a vast area in PSI and it was grown historically in an unorganized way. The miscellaneous network hardware infrastructure and the lack of the documentation and components overview could no longer guarantee the reliability of the control system and the facility operation.

Therefore, a new network, based on modern network topology, PSI standard hardware with monitoring and detailed documentation and overview was needed. We would like to present how we successfully achieved this goal and the advantages of the clean and well documented network infrastructure.

INITIAL SITUATION

The HIPA control system network components and nodes are located in about six buildings. It expanded during many years, often with ad-hoc requirements. It consisted of about 25 network switches of various manufacturers, 150 computer nodes and 20 operator consoles. Numerous problems started to arise with ever increasing frequency.

The lack of general HIPA control system network overview made the system administration difficult. Problems with performance, such as bottlenecks during data transfer between the private machine network and the PSI office network, led to the situation when no central backup and operating system update, by using the PSI standards, were possible.

Further there were problems, based on the network topology. A cascade topology rather than a star topology had evolved over the years.

Another problem we encountered, related to the maintenance of network components. In particular, switches have been installed from a multitude of suppliers and so it was almost impossible to install replacements from our emergency stock. This was a risky situation for the facility operation.

And finally the network administration was not satisfactory. Central monitoring was not possible at all, because most of the network components did not support management or monitoring.

The original situation can be seen on Fig. 1. It shows switches of various manufacturers, often connected in cascade, in six buildings. This diagram has been created after a detailed analysis of the current situation at the facility during November 2010. Fig. 2 illustrates cascaded production switches from various manufacturers used until the end of 2010.

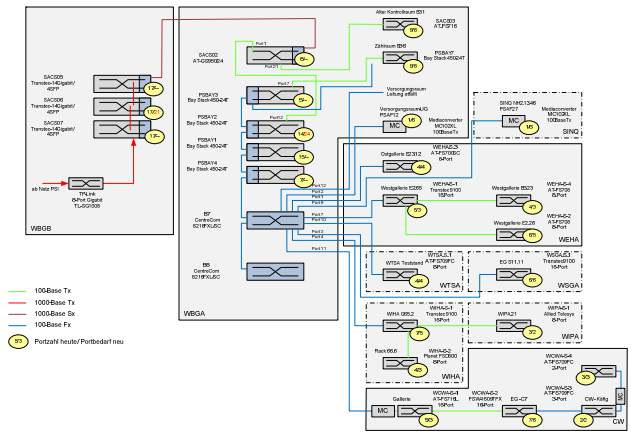


Figure 1: The HIPA network overview after the current status analysis in November, 2010.

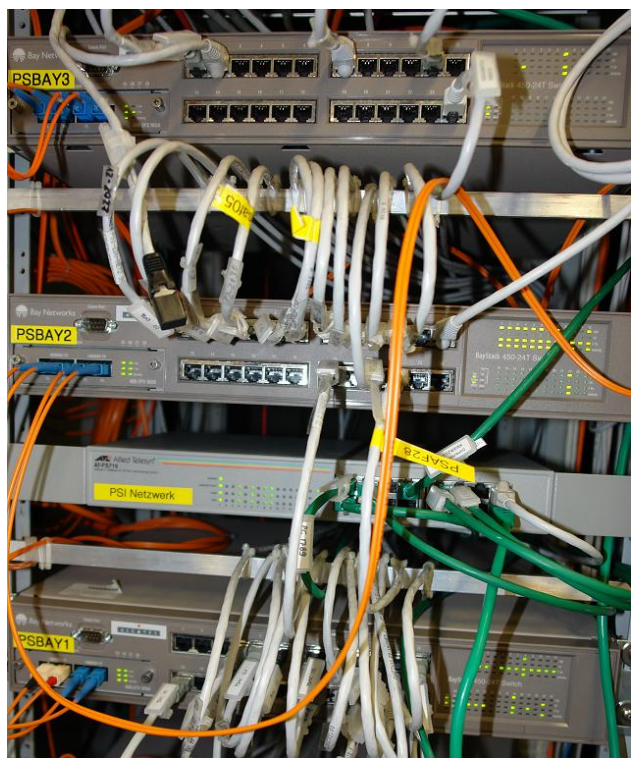


Figure 2: Different flavour of network switches running in HIPA Controls network until the end of 2010.

PROJECT GOALS

The main goal for the HIPA network migration was to build a highly available, stable and scalable network with low complexity and dependencies. This should mainly lead to the improvement of network performance. Furthermore, network administration and hardware maintenance should be more simple and effective. Network design topology should be according the PSI standards with network monitoring. The project should be well documented in order to supply overview and useful information for the facility operation or for people on the on-call service.

PROJECT PLAN AND REALISATION

Three teams collaborated intensively during the project preparation and realisation: At PSI these were the Controls and AIT Network groups and the external company KeyNet Consulting GmbH [1].

Based on the analysis of the current situation, a new design based on the star topology was created, see Fig.3.

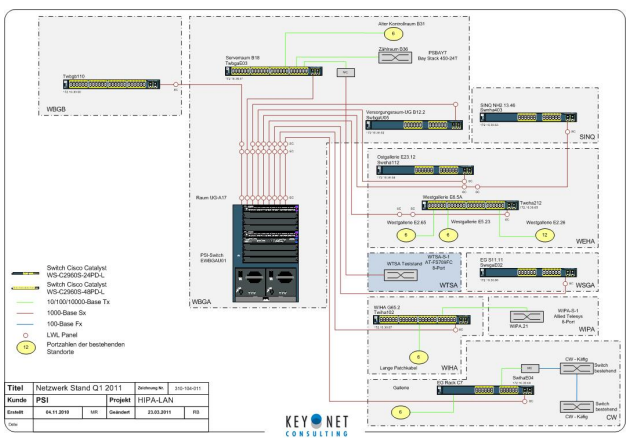


Figure 3: The new network design based on the star topology, PSI standard.

Afterwards, a more detailed project plan was elaborated. This included a schedule for ordering and purchasing of switches and other components; and cabling specification for the Electrical engineering division.

The cleanup of nodes in the new and old server rooms and control rooms has been done. The configuration of operator consoles, servers, and other computing components has been unified. Redundant computers have been removed.

One of the biggest challenges of the network realisation was the tight time scheduling during the HIPA shutdown period. An important part of the project was the cabling realisation. Because of minimal resources, it was difficult to schedule the network recabling to the electrical shutdown plan. All the activities had to be independent of other shutdown activities, such as components and machine tests. And finally, the successful operation start-up schedule had to be guaranteed. Another challenge was the fact that we could enter some locations only for a very short period because of high radiation. This was for example, the building in which the ion source and the Cockcroft-Walton accelerator are located.

RESULTS AND ACHIEVEMENTS

The HIPA proton accelerator control system runs now on a modern scalable and stable PSI standard network.

Network Homogeneity

The number of active components has been reduced from 25 to 9 Cisco Catalyst 24- or 48-port switches. They are the same type as other PSI switches, thus a replacement emergency stock is not an issue anymore.

Performance

We could observe performance improvement, thus the computers can receive the PSI updates and servers are connected to the PSI central backup system.

DHCP and DNS

DHCP and DNS administration has been reorganized by using the PSI standard infrastructure.

Cabling

Cabling and patching work has been done meticulously. Many unused connections have been unplugged and hundreds of meters of cables removed. This led to a better physical overview.

Monitoring

The HIPA network is now monitored by using the PSI standard tool, NeDi (Network Discovery) [2] and the information is visible to the Controls section on the intranet, see Fig. 4.

Documentation

Even the documentation has been written during the each process of the HIPA Network Upgrade, we realized on-line documentation generated directly from the Oracle database. All the information was updated in the Controls Database Inventory [3]. The tool generates various reports in the form of maps and diagrams working directly with the data stored in the database. They can be displayed on the PSI intranet Web, see Fig. 5.

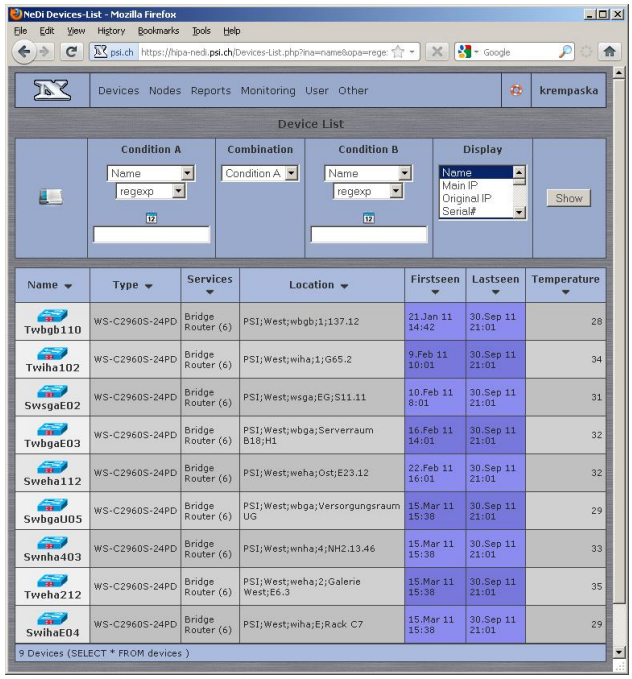


Figure 4: HIPA switches and nodes are monitored by a PSI Network monitoring Web tool NeDi.

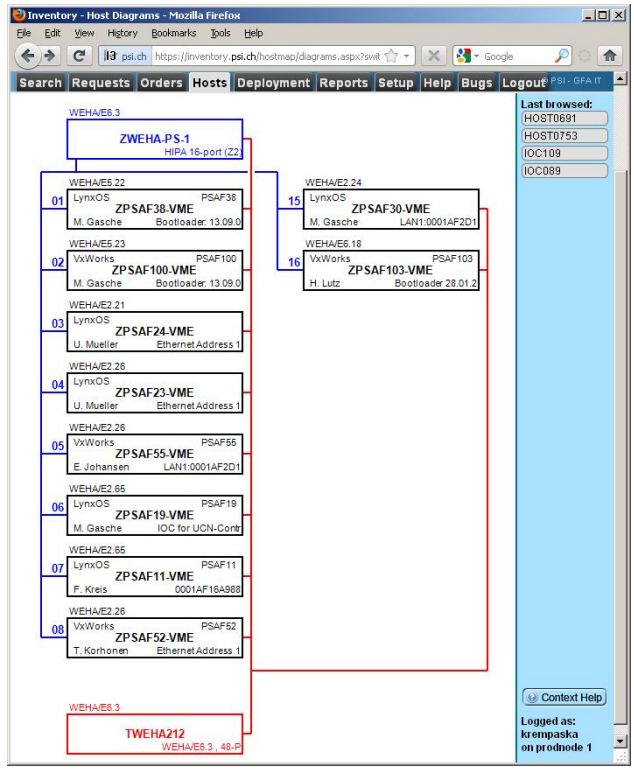


Figure 5: The diagram shows a switch and the nodes connected to it (red lines). These VME IOCs have serial connections to a port server (blue lines).

ACKNOWLEDGMENTS

We want to acknowledge M. Rutz and R. Boesch, KeyNet Consulting GmbH, Luzern, Switzerland for the project management consulting, documentation and technical support. Further, we wish to thank to the colleagues from PSI Infrastructure and Electrical Engineering Divisions for cabling specification and realisation, especially to R. Kaech.

REFERENCES

[1] R.Krempaska, H. Lutz, F.Lendzian, PSI, Villigen, M.Rutz, R. Boesch, KeyNet Consulting GmbH, Projektabschlussbericht Redesign HIPA-Network
[2] http://www.nedi.ch/
[3] http://gfa-it.web.psi.ch/invent_help/

DETECTOR CONTROL SYSTEM OF THE ATLAS INSERTABLE B-LAYER

S. Kersten, P. Kind, K. Lantzsch, P. Mättig, C. Zeitnitz, Bergische Universität Wuppertal, Germany
F. Gensolen, CPPM, Marseille, France; M. Citterio, C. Meroni, INFN Milano, Italy
B. Verlaat, NIKHEF, Amsterdam, Netherlands; S. Kovalenko, PNPI, St. Petersburg, Russia

Abstract

To improve tracking robustness and precision of the ATLAS inner tracker, an additional, fourth pixel layer is foreseen, called Insertable B-Layer (IBL). It will be installed between the innermost present Pixel layer and a new, smaller beam pipe and is presently under construction. As, once installed into the experiment, no access is possible, a highly reliable control system is required. It has to supply the detector with all entities required for operation and protect it at all times.

Design constraints are the high power density inside the detector volume, the sensitivity of the sensors against heat-ups, and the protection of the front end electronics against transients. We present the architecture of the control system with an emphasis on the CO_2 cooling system, the power supply system, and protection strategies. As we aim for a common operation of Pixel and IBL detector, the integration of the IBL control system into the Pixel control system will also be discussed.

DETECTOR LAYOUT

The Insertable B-Layer is a pixel detector, which is mounted directly on the LHC beam pipe. Fourteen mechanical support structures, called staves, form a cylindrical shell with an average radius of 33.25 mm and a length of 698 mm. A pseudo-rapidity of $|\eta| < 3$ is covered. Each stave carries sixteen modules. They are composed of the sensor itself and two frontend chips, whose read out cells are bump bonded to the sensor cells. While the inner modules of each stave will be built by planar sensors, the outermost modules of each stave are 3D sensors. In total there are three quarters of planar and one quarter of 3D sensors. For details of the different sensor technologies, see [1]. The frontend readout chips, the FE-I4s, are produced in a 130 nm bulk CMOS process. Its pixel cells are organized in 80 columns with a pitch of 250 µm and 336 rows with 50 µm pitch. Altogether 448 frontend chips provide 12 million readout channels.

Due to the high power dissipation of the detector modules, an efficient cooling system is required. Therefore a cooling pipe is integrated into each stave. An evaporative cooling using CO_2 keeps the detector modules at (-25 to -30) °C. A total power of 1.4 kW can be removed from the detector. Extra cooling power is available to remove the environmental losses of the transfer piping.

The data of the frontend chips are sent to the counting room via optical data transfer. In the other direction configuration, trigger and control signals are sent by the DAQ system to the frontend chips, also using the optical link. The on detector part of the opto-electrical transceivers, the opto boards, are located close to the detector itself, while the off detector part, the Back of Crate (BoC) cards are installed inside the DAQ crates.

DCS REQUIREMENTS

The Detector Control System (DCS) is responsible for the safety of human beings, the detector and other equipment. It must provide tools to control the detector in all operation modes, like normal data taking with colliding beams, as well as calibration and tuning periods. To ensure the success of a command, feedback must be given to the operator. Furthermore the DCS must deliver additional information for debugging the detector and diagnosing its behaviour. Safety, control and diagnostics vary concerning required reliability, availability and granularity. Highest level of reliability is required for the safety system, a high level for all control actions, while the level can be lower for information which is used for diagnostics. Similar levels must be considered concerning the availability of a DCS component. While the safety system must be functional at all times, control units are required as soon as an operation is performed with the detector and diagnostic tools are activated on request. Concerning the modularity the levels are reverse. If the behaviour of the detector must be analysed, detailed information on small units like individual frontend chips must be available. On the other hand, if the detector must be protected against upcoming risks, even larger parts of the detector can be stopped.

A clear user interface, automatic safety and error recovery procedures and archiving of data are tools to fulfill the above listed requirements. Furthermore the integration of the IBL DCS into the Pixel and the ATLAS wide control system [2] must be ensured. These demands have impact on the choice of hardware as well as on the software design of the IBL control system.

DCS HARDWARE

An overview on the IBL DCS hardware is given in Figure 1. The detector modules, the opto boards and the BoC/DAQ crates are subjects to DCS. Furthermore the detector environment must be permanently watched. These are the entities which require monitoring and/or control.

Several power supplies (HV, LV, SC-OL) provide all units with the power they need. On the other side the high power dissipation is compensated by the cooling system.

Various monitoring mechanisms are in place, some of them provide in parallel inputs to the interlock system. This system is responsible for the safety of human being and of the detector and acts directly on the power supplies if an error condition is found. The normal control and monitoring functions are provided by the Cooling and DCS PCs. Most parts of the DCS are located in the counting room which is about 100 m away from the inner detector itself. Just the regulator station and some monitoring units are installed inside the detector cavern.

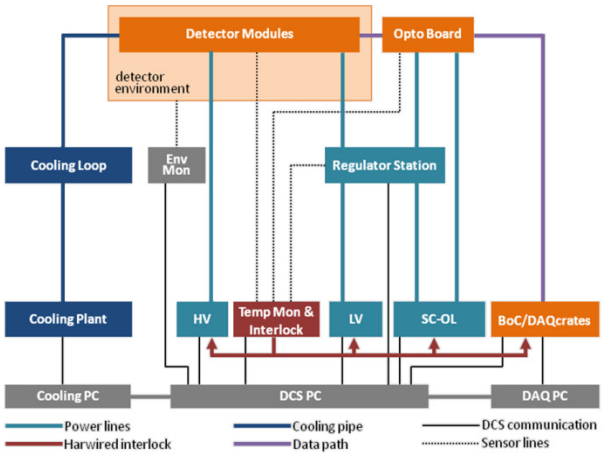


Figure 1: DCS Overview.

Power Supplies

Detector modules and the opto boards require a dedicated powering which is exactly adapted to the needs of the loads. To all components in common is the request of floating supplies. The output voltages must be variable. Concerning safety, over-current protection and interlock inputs are of main importance.

To deplete the sensors a HV power supply is required. While the planar sensors will need up to 1000 V after irradiation, the 3D sensors can be operated with significantly lower voltages of a few hundred Volt.

The frontend chips are powered by the low voltage (LV). Due to the required currents of up to 900 mA per frontend chip the voltage drops on the services are not negligible. The frontend chips themselves require just 1.2-1.5 V, however the LV supply will be able to deliver 10 to 15 V. As on the other side the frontend chips, which are produced in a deep submicron technology, would be destroyed by over voltages of more than 4 - 5 V, a voltage regulation close to the detector is performed. It protects the sensitive chips against transients and is therefore an important safety mechanism. Key components of the regulator station are the regulator LHC4913 and digital trimmers which allow for a remote control. For more details on the regulator station see [3].

The third voltage supply SC-OL (supply and control of the opto link) is responsible for the powering of the opto boards, which require three different low voltages [4]. The VCSEL driver chip and the decoder chip are powered by Vvdc with a maximum voltage of 10 V and 800 mA. As also the chips of the opto link must be protected, Vvdc is routed through the regulator station in the same way as LV. The receiver diodes need a higher voltage of up to 20 V, while 5 V are sufficient for Viset. Viset allows for control of the VCSEL output current. Additionally a reset signal is provided which can be sent to the decoder.

Cooling System

The 14 IBL staves are cooled with 14 parallel cooling loops with evaporative CO_2 of around -40 °C. The 14 cooling loops have manifolds in the muon detector area which is accessible in technical stops. The cooling pipe in the stave has an inner diameter of 1.5 mm and is made of titanium. The in and outlet tubes are both 12 m long and have an increasing diameter in the upstream direction to accommodate the expanding vapour. The inlet is 1 mm and 1.5 mm, the outlet 2 mm and 3 mm.

The cooling plant is located in the USA-15 cavern, the piping distance to the detector is about 100 m. The cooling system is using the 2PACL cooling concept [5] as it was developed for the LHCb-Velo cooling [6]. In a 2PACL system the pressure of boiling is controlled by an accumulator in the plant. The accumulator pressure is controlled by heating or cooling. The pressure in the detector is similar to the plant pressure and thus can the boiling pressure and hence temperature can be controlled from the plant only. With this system no active control and sensing is needed inside the unaccessible detector. The cooling inside is completely passive. Temperature sensors for monitoring are available on each stave in- and outlet. Figure 2 shows a schematic of the IBL CO_2 cooling. Half of the amount of staves will have a flow in the opposite direction so that the outlet flow can cool the inlet flow of the other tubes. This pre-cooling is needed to suppress boiling in the inlet due to parasitic heat. The cooling loops are connected to the transfer line in the junction box located in the muon detector area. The transfer line is a concentric pipe exchanging heat between the in and outlet. This heat exchange conditions the inlet flow to be the right temperature for boiling. As all the pipes are cold, they have to be insulated. Inside the detector there is vacuum insulation, outside the detector there is foam insulation. Two identical cooling units are foreseen for redundancy reasons.

Monitoring

While all power supplies have built-in monitoring units which provide information concerning voltage and current, the temperature monitoring is handled separately. Many temperature sensors are installed and deliver their information into DCS through different paths.

Any equipment which can be damaged by overheat is equipped with an NTC (Negative Temperature Coefficient) sensor, whose value is read by the DCS monitoring units. In parallel its information is fed into the interlock system (see the next section). Further NTCs on the cooling pipes, the cable trays, and elsewhere in the detector environment provide the cooling experts with information which allows for debugging of the system.

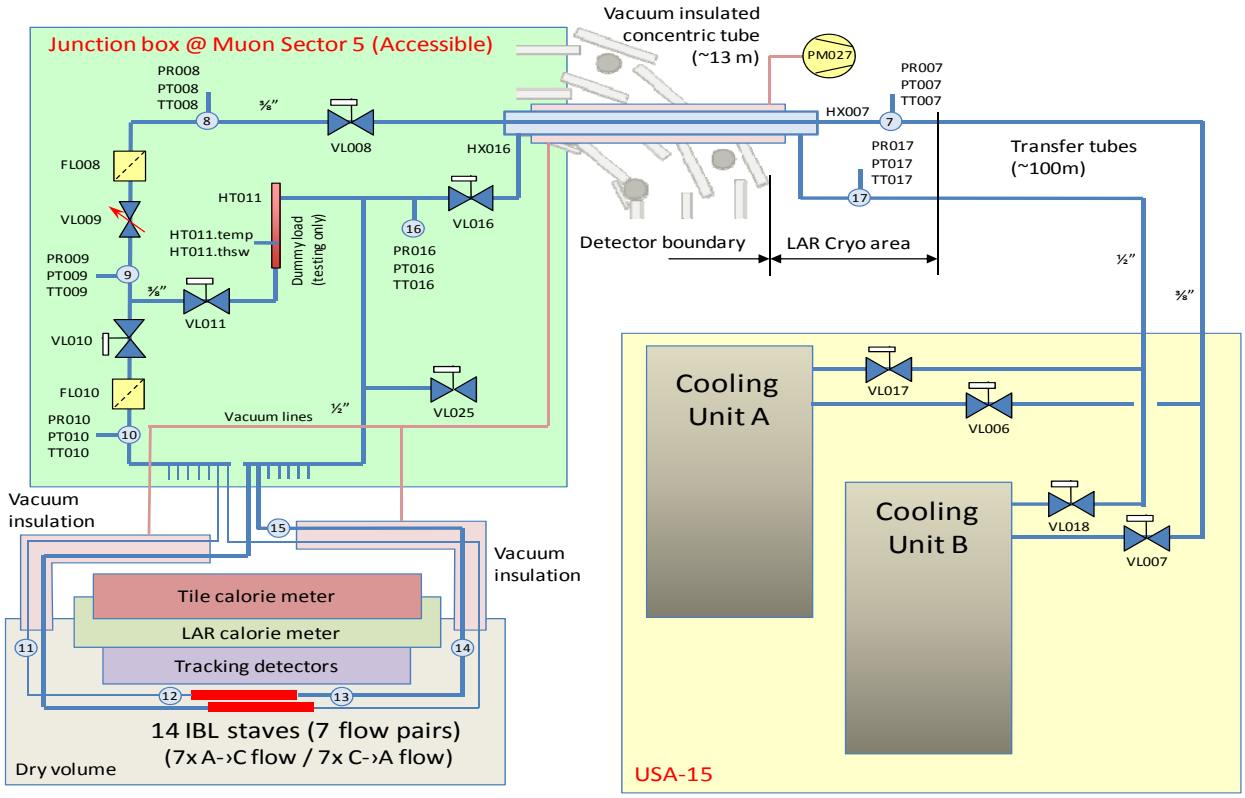


Figure 2: Cooling System.

A further source for detailed debugging of the frontend chips are, among others, temperature sensors (diodes), which are built-in in the frontend chips themselves. In order to control the complex electronics of the FE-I4 [7] and its environment, several voltages and currents are monitored on-chip and digitized for readout. A 10 bit ADC has been designed for this purpose and is associated to an 8 to 1 analog mux in order to select one input among: temperature, power supplies, voltage references, detector leakage current, and other DC analog voltages. On demand these information can be sent through the standard data path. Inside the DAQ dedicated routines extract the DCS information. This monitoring is a promising approach for future Detector Control System developments in order to reduce the material inside the detector volume.

As the monitoring of the DAQ crates and BoC cards is based on standard monitoring tools, it is not described here.

Interlock

As irradiated sensors can be irreparably damaged by overheat, the detector modules are equipped with NTCs which are readout through DCS and in parallel feed their information into the interlock system. A discriminator compares the signal to a threshold which represents the maximum allowed temperature, and creates a logical signal in case of overheat. In the same way, opto boards and the regulator stations are equipped with NTCs.

All signals are collected by the interlock system. It determines which power supply must be switched off and sends out the related signal to the power supply. The core component of the interlock system is a flash FPGA with an internal EEPROM. This avoids the need of a program loading at power-on. A negative logic is implemented, which means that a missing cable or power loss cause automatically an interlock. Additional information from the laser protection system, the cooling system, the LHC or other external systems will be included into the interlock matrix.

As this system is completely hardware based and does not require any software to be running, it provides a maximum of safety.

DCS SOFTWARE

The DCS software splits into three parts, one handling the integration of the hardware into the software, the user interface to operate the detector and connections to the databases plus tools for monitoring.

It is foreseen to purchase the power supplies from the same vendors as they were selected for the Pixel detector and to use the same hardware for custom made devices

(e.g. regulator station, SC-OL). Therefore, the pixel software packages can also be applied to integrate the IBL hardware. Besides the fact that large development work can be avoided, using the same integration software also guarantees an easy integration of the IBL DCS into the pixel frame.

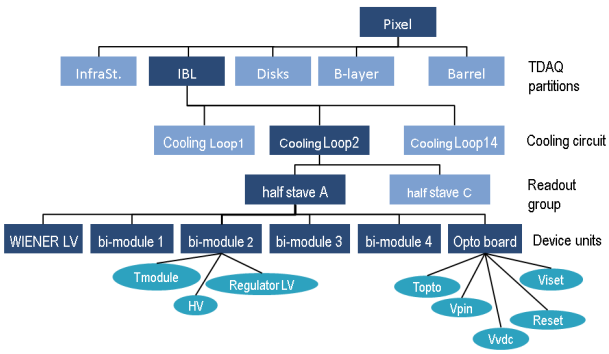


Figure 3: FSM tree of the IBL.

The tool to operate the detector and all its components is the FSM (Finite State Machine). Using the control concept as it is defined by the ATLAS FSM [8] provides the integration of the IBL into the pixel detector and ATLAS. Figure 3 shows the FSM tree of the IBL. The top nodes (TDAQ partitions) are defined by ATLAS. It is also evident that the IBL is a part of the pixel detector. The nodes below IBL reflect the detector's constraints. As done for the pixel detector, a geographical approach is followed.

As the cooling loops can be controlled separately, they define the next level of partitioning. They correspond each to one stave. All modules of a half stave share the same opto board concerning the readout. This defines the next level of partitioning. Besides the opto board, LV and four bi-modules belong to a half stave, respectively readout group.

Due to the services four frontend chips with their two sensors (in case of planar sensors) are the smallest unit which can be separately steered by DCS and are called bi-module.

If the detector behaviour must be studied in more detail the 3D-viewer is a useful tool for experts. It is based on the fw3DViewer package (developed by CERN ITCO [9]) and allows for monitoring of any data available with module granularity. Online as well as historical data can be displayed in any 3D view. Especially 3D views of temperatures or leakage currents can help to diagnose problematic detector regions.

SUMMARY

In 2013 the IBL will be installed as the innermost part of ATLAS. It will complement the existing Pixel detector as a fourth layer. Although being a pixel detector, its sensors and frontend chips are different concerning their qualities and needs.

A complex detector control system is required to supply all required voltages, to keep the detector at an adequate operating temperature, to provide operators and experts with all tools and information they need, and to ensure the safety of the detector at all times.

Key component regarding safety is the interlock system. Furthermore, the sensitive frontend electronics is protected against transients by the regulator station. The standard tool to control the detector will be the finite state machine, which in parallel allows for an easy integration of the IBL DCS into the Atlas wide control system. For debugging the detector and diagnosing its behaviour the built-in sensors of the frontend electronics and the 3D viewer are of special interest.

REFERENCES

[1] ATLAS Insertable B-Layer, TDR, CERN-LHCC-2010-013
[2] S. Schlenker et al., "The ATLAS Detector Control System", these proceedings
[3] http://atlas-pixel.mi.infn.it/wiki/index.php/PP2_regulators_and_boxes
[4] http://hepweb.physik.uni-wuppertal.de/index.php/detlab/dcs/89-dcs-fuer-den-ibl.html
[5] Verlaat B., "Controlling a Two-Phase CO2 Loop Using a Two-Phase Accumulator", ICR07-B2-1565, International Conference of Refrigeration, Beijing, China, 2007
[6] Verlaat B. et al, "CO2 Cooling for the LHCb-VELO Experiment at CERN", CDP 16-T3-08, 8th IIF/IIR Gustav Lorentzen Conference on Natural Working Fluids, Copenhagen, Denmark , 2008
[7] D. Arutinov et al., "FE-I4 Chip Development for Upgraded ATLAS Pixel Detector at LHC", Pixel 2010, 5th International Workshop on Semiconductor Pixel Detectors for Particles and Imaging, September 6 – 10, 2010, Grindelwald, Switzerland
[8] A. Barriuso Poy, S. Schlenker, FSM Integration Guideline, ATL-DQ-ON-0010
[9] http://j2eeps.cern.ch/wikis/display/EN/JCOP+Framework+3DViewer

LHC MAGNET TEST BENCHES CONTROLS RENOVATION

O.O. Andreassen, D. Kudryavtsev, S. Page, A. Raimondo, A. Rijllart, E. Zorin
CERN, Geneva, Switzerland

Abstract

The LHC magnet test benches controls were designed in 1996. They were based on VME data acquisition systems, Siemens PLCs control and interlocks systems. After a review of renovation of superconducting laboratories at CERN in 2009, it was decided to replace the VME systems with a PXI based systems and the obsolete Sun/Solaris workstations with Linux PC's. This paper covers the requirements for the new system and shares the experience of the upgrade of the magnet test benches to these new platforms.

OVERVIEW

A review of superconductors and magnet laboratories took place in May 2009 [1]. It concluded that the removal of the facilities of Block4 to SM18 [2] was desirable to save money, to stop the Block4 refrigeration plant and to increase of efficiency by staff regrouping. Another important reason was related to the FRESCA2 magnet for the superconducting cable test, which was not possible to be installed in Block4 due to the lack of space.

Merging the two facilities allowed the creation of an integrated magnet test facility, adding to the existing horizontal test benches a vertical test station for prototype magnets, a test bench for the existing insertion region magnets, as well as a test station for the new generation inner triplet magnets presently under design.

The ageing infrastructure, the computer systems and the instrumentation needed to be modernised and the controls of the different systems of the two magnet test facilities to be homogenised.

This paper describes the Magnet Test Benches (MTB) controls renovation project. The project targets the Power Converter Controls and the Measurement Systems. A common objective is to bring them as close as possible to the systems presently installed in the LHC to make the MTB look like the "ninth" LHC sector (the LHC has 8 sectors) from the control system point of view. This is important for tests such as the tracking measurements. This effort is complemented by the migration from the General Purpose Network (GPN) to the accelerator Technical Network (TN) and the implementation of the Role-Based Access Control (RBAC) to increase cybersecurity.

THE NEW MTB FACILITY

The new MTB facility (Fig. 1) was created in three main steps.

First, the removal of the vertical test station from the Prévessin site to the SM18 site; second, the modification of one horizontal test bench in order to allow testing the spares of the present inner triplets and other magnets that were produced and tested by two US laboratories; finally the modification of one horizontal test bench to allow the testing of the new inner triplet prototype magnets. These three different parts are independent and can be realized at different times.

The removal was also the opportunity to upgrade the vertical test station. It was equipped with 4 cryostats of different dimensions for different purposes: 3 of them were used in the previous facility and a 4th, the biggest, was procured within the High Field Magnet (HFM) project for the FRESCA2 magnet.

These cryostats allow to test the present LHC corrector magnets, diodes for LHC main dipoles and quadrupoles, new inner triplet correctors, new inner triplet quadrupole model, and the HFM prototypes. With some modification of one cryostat, we should be capable to test also the present LHC HTc current leads.

The vertical test station will also allow the test of Fast Cycled Magnets (FCM) and Superconducting Links (Sc link) for the new inner triplet and FP7 (EUCARD) project.

A 20 kA and 25 V power supply from the former facility have been recovered for the new station together with several 600 A and +/- 12 V power supply to assure the powering of the present LHC and future new inner triplet magnets. The existing commutation switch of 20 kA completes the installation.

This new station profits from the cryogenic installations and He gas recuperation system of the test area.

The FCM and the Sc link, not operating at the same time, will use a common cryogenic distribution box, existing in the hall.

A modification of the liquid distribution, gas recuperation and the control system of the cryogenics of the new station has still to be done.

The former protection and detection systems and the data acquisition system of the vertical test stand were obsolete and could not be integrated into the new system. Therefore a new one had to be developed and installed in the same time with the new station.

The horizontal benches to test the LHC main magnets, such as dipoles and corrector magnets, will now share optimally the resources with the vertical benches.

Figure 1: Part of the merged MTB facility.

Power Converter Control Upgrade

Every power converter is controlled by a Function Generator Controller (FGC) that hosts the control software. Normally a set of FGCs is connected to the Control Middleware (CMW) [3] via a gateway PC through a WorldFIP bus.

The upgrade of the FGC gateways was a relatively simple operation. The operating system needed to be switched from LynxOS to SLC5 Linux, and the start-up sequence had be updated to be identical to the one used in the LHC.

Following the upgrade, commands are sent through the CMW and must be accompanied by a valid Role-Based Access Control token [4]. RBAC rules granting access to the power converters have been defined. A new RBAC role and RBAC location has been created and is populated with the names of the new Linux controls consoles.

The majority of the power converters were controlled by old FGC1 hardware. In order for the FGC1s to work in conjunction with the new version of the gateway, a minor upgrade has been made to their software.

The first step to integrate the MTB power converters into the standard LHC architecture was to connect them and their control console to the Technical Network. This meant upgrading the obsolete network, removing the coaxial cables to be replaced with optical fibres and replacing the old star points. At the same time the objective was to improve the actual network (WiFi covered zones, GPN or TN, additional sockets for the office spaces) and to add new network sockets for the relocation of the test components.

This migration from coax GPN to structured TN network of all the devices has been done in a staged way to avoid stops in the MTB operation.

The upgrade of the building gateways was needed for the LHC software architecture (LSA) to directly control the power converters. LSA can directly control the FGC2 interfaces and with minor changes in the name of the properties the FGC1 interfaces. This additional flexibility will probably be needed in the future.

Once done, the existing direct current readout from the FGC1 used by the old measurement applications would not be available anymore and the control applications had to be upgraded.

The FGC2 interface is requested in order to have the "post mortem" data with the detailed executed current cycle or to investigate eventual problems offline in detail. Both systems (gateway and FGC) are synchronised with the Universal Time (UTC) as foreseen for all the accelerator systems at CERN, with a precision of 1 ms, that is considered necessary and sufficient.

Last but not least, it is worth to have the last electronic version (i.e. FGC2 interface) to control all converters for maintenance reasons and for the ongoing magnet control developments.

Measurement Systems Upgrade

The first step of the upgrade was to replace the existing VME based Data Acquisition (DAQ) system with a new one based on PXI hardware and LabVIEW RT.

Previously each cluster was equipped with an INCAA DAQ system for magnet power tests. These systems were getting old and the SUN Solaris operating system, on which the control consoles were based, has been phased out. Moreover LabVIEW was no longer updated on this operating system. The data acquisition systems used previously had to be renewed as the installations were 20 years old and were not reliable and adapted for the test of the new inner triplet magnets.

Therefore the installations had to be changed, adapted to those used at CERN presently.

A new architecture has been defined in our group (Fig. 2) and a prototype based on PXI has been built with the goal to test it on present LHC magnets and to qualify the solution to be applied on each horizontal bench and the vertical test station.

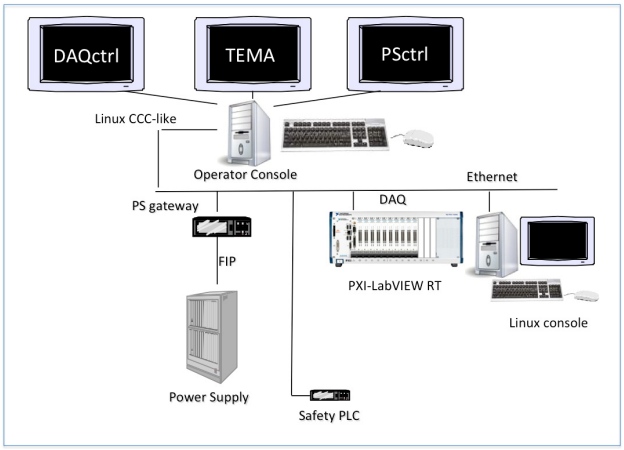


Figure 2: The control architecture.

The second step was to port the actual Test Master, the application that drives the sequence of tests, and Hardware Recognition, the application that allows to configure the electronics, on Linux with the addition of a DAQ control application.

The next step was to port the actual Power application on Linux to access all the Power Converters via the LHC Control Middleware using RADE [5] from the control room. The new control room has become similar to the CERN Control Room.

On the three screen consoles; the middle one shows the Test Master window, which is equivalent to the sequencer for the LHC Hardware Commissioning, and on the left and right screens there are the DAQ and the Power Converter control applications.

CONCLUSIONS

The review of superconductors and magnet laboratories concluded the removal of the facilities of Block4 to SM18 site. Merging the two facilities allowed the creation of an integrated magnet test facility, adding to the existing horizontal test benches a vertical tests station for prototype magnets.

Thanks to the merging we started a renovation program to replace the obsolete VME DAQ and Solaris consoles with modern PXI RT and Linux consoles. In addition the update of the FGCs has made the MTB look like the "ninth" LHC sector from the control system point of view.

The renovation of the control architecture started end of 2009 and involved 3 FTE up to now. The upgrade went in parallel with the standard magnet test benches operations to keep the compatibility between the new and the old systems as they needed to run in parallel.

The upgrade has now been completed and all the test benches are now commissioned.

Further improvements are ongoing until the end of this year.

REFERENCES

[1] "Internal review of superconductors and magnet laboratories" http://indico.cern.ch/conferenceDisplay.py?confId=56351 chaired by L. Bottura and L. Walckiers, 2009.

[2] "Cold tests of the MB cold masses in SM18", L. Walckiers, 1st LHC Project Workshop, 2004.

[3] K. Kostro, "The control middleware (CMW) at CERN status and usage", ICALEPCS2003, Gyeongju, Korea.

[4] "Role-Based Access Control for the accelerator control system at CERN", S. Gysin et al., Proceedings of ICALEPCS07, Knoxville, Tennessee, USA, 2007.

[5] A. Raimondo et al., "Rapid Application Development Environment based on LabVIEW", CERN, Geneva, 2008, http://cern.ch/rade

EVOLUTION OF THE ARGONNE TANDEM LINEAR ACCELERATOR SYSTEM (ATLAS) CONTROL SYSTEM*

M. Power, F. Munson, Argonne National Laboratory, Argonne, IL 60439, U.S.A.

Abstract

Given that the Argonne Tandem Linear Accelerator System (ATLAS) recently celebrated its 25th anniversary, this paper will explore the past, present, and future of the ATLAS Control System, and how it has evolved along with the accelerator and control system technology.

ATLAS as we know it today, originated with a Tandem Van de Graff in the 1960s. With the addition of the Booster section in the late 1970s, came the first computerized control. ATLAS itself was placed into service on June 25, 1985, and was the world's first superconducting linear accelerator for ions. Since its dedication as a National User Facility, more than a thousand experiments by more than 2,000 users world-wide, have taken advantage of the unique capabilities it provides.

Today, ATLAS continues to be a user facility for physicists who study the particles that form the heart of atoms. Its most recent addition, CARIBU (Californium Rare Isotope Breeder Upgrade), creates special beams that feed into ATLAS.

ATLAS is similar to a living organism, changing and responding to new technological challenges and research needs. As it continues to evolve, so does the control system: from the original days using a DEC PDP-11/34 computer and two CAMAC crates, to a DEC Alpha computer running Vsystem software and more than twenty CAMAC crates, to distributed computers and VME systems. Future upgrades are also in the planning stages that will continue to evolve the control system.

ATLAS' FIRST CONTROL SYSTEMS

In 1978 a proposal was submitted to create a super-conducting LINAC using seven groups of independently-phased split ring resonators, and called for a computer based control system. The proposal was to use a Digital Equipment Corporation (DEC) PDP 11/34 to "(1) set and monitor all parameters of resonators and solenoids, and to control long-term drifts, if any; (2) to monitor many sensors such as thermometers; (3) to automate the tuning operations" [1]. Figure 1 shows the original proposed design, using two CAMAC crates, two touch screen panels integrated with knob controls, and a numeric keypad as shown in Figure 2, and a raster display as operator interfaces. Control programs were written mostly in FORTRAN on the RSX/11 operating system. Some of this code is still in use today.

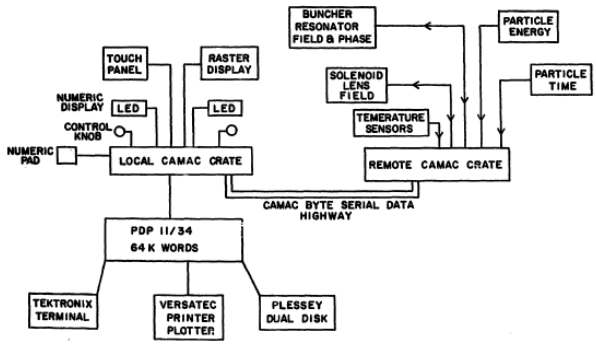


Figure 1: Original computer control system proposal.

Figure 2: Touch screen control board.

As the accelerator continued to expand, so did the control system to include three PDP 11 systems, each controlling part of the accelerator: the newly constructed Positive Ion Injector (PII), the primary accelerator section (Booster and ATLAS), and beamline devices. All three systems contained separate and isolated databases. More CAMAC hardware was added and separated into two serial highways [2]. Figure 3 shows the control room after the expansion.

* This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under Contract No. DE-AC02-06CH11357.

Figure 3: Control Room circa 1985. The right (grey counter) is the accelerator control system including the original Booster and added ATLAS sections. The left (white counter) is the added beamline control system.

MAJOR UPGRADE

In the early nineties, plans to update the control system to create a more integrated and expandable system using newer technologies began to be discussed. Since there was a significant investment in the existing CAMAC hardware, it was decided to retain it, but the crates were reconfigured into a single serial highway. The multiple PDP 11 computers were replaced with a single DEC MicroVAX running VMS software, and an Ethernet based LAN was installed to interconnect the various display computers with the server. A third party software package, Vista Control Systems Vsystem, was chosen to provide an integrated database, graphical user interface, and hardware support [3]. Figure 4 depicts one of the new Vsystem displays.

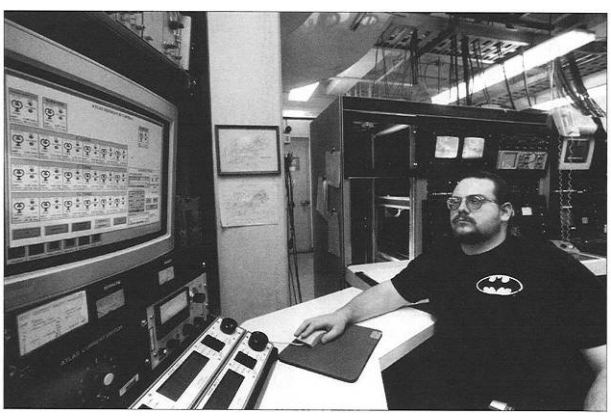


Figure 4: New Vsystem display in the Control Room.

Gradually, new pieces were phased into the system. The software written for the PDP 11 computers was transferred to the MicroVAX. Once the port was complete, a new DEC AlphaServer was purchased and replaced the MicroVAX. The AlphaServer was the core of the control system and contained the single link to the CAMAC serial highway and all the real time databases and displays. Various other Alpha workstations and PCs were used as operator workstations throughout the accelerator site. Figure 5 shows the conceptual layout of the upgraded control system.

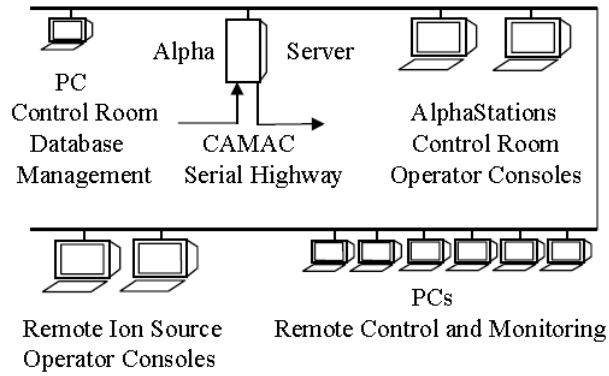


Figure 5: 1990's upgrade configuration.

Additionally, two relational databases were added to provide logging of system parameters and the ability to scale and reload control parameters into the real-time system for similar charge to mass ratios. This was accomplished by storing real-time data on the AlphaServer using Oracle RDB, and periodically transferring the data onto a remote PC located on the LAN. The PC software was written with Corel Paradox to store the data and provide an interface for the operations staff.

Another noteworthy accomplishment of this upgrade was that it was done for the most part online, simultaneous with the operation of the accelerator. The ported software control programs including Resonator and Solenoid Control, Autoscan, and Energy and Time Measurement applications are still in use today.

DISTRIBUTED SYSTEMS

As ATLAS would continue to grow, and with the understanding of the limitations of having a single point of failure with a single server and single CAMAC serial highway, efforts were made to implement a distributed system test case. The Vsystem software already supported distribution, and it was decided to move the cryogenics database to a distributed system. The cryogenics database and its related displays were copied to a PC with the Linux operating system running the Vsystem software. A PCI to CAMAC interface was installed, and the CAMAC crate was removed from the main serial highway and connected directly to the Linux PC. Figure 6 shows the new configuration [4].

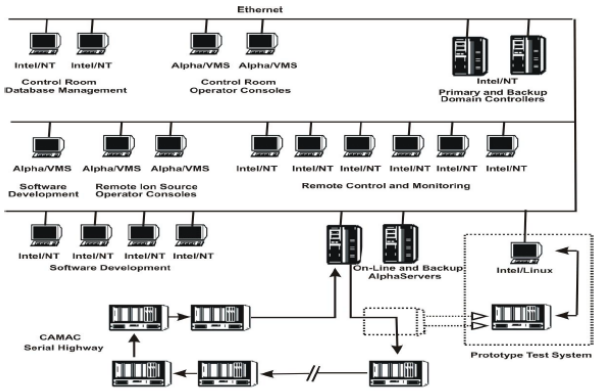


Figure 6: ATLAS control system with distributed I/O.

More recent accelerator upgrades including the new Charge Breeder and the addition of the CARIBU (Californium Rare Isotope Beam Upgrade) Source have increased the use of distributed processing. With increasing prices due to the low demand for CAMAC modules, additional hardware options have also been investigated. Two hardware options using Industry Packs have now been implemented. Four VME crates have been installed, and are controlled using Linux based PCs running the Vsystem software. A second option was initially installed on the source high voltage platforms. Two Hytec 9010 blade I/O controllers [5] were installed using the onboard Industry Pack support. These controllers also run the Vsystem software on Linux, and are excellent in areas where less I/O is needed. Due to problems with the 9010s caused by an occasional electromagnetic pulse due to high voltage arcing in these locations, the 9010s have been replaced with VME systems, but will be repurposed for use in other areas as the system continues to be distributed and offloaded from the DEC AlphaServer. Figure 7 shows the current control system configuration.

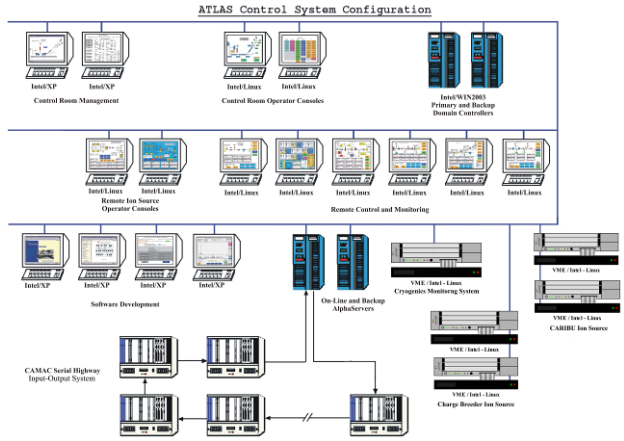


Figure 7: Current ATLAS Control System configuration.

FUTURE PLANS

ATLAS continues to evolve and expand, meeting the future needs of its users. Additional upgrade projects are currently underway and the control system will continue to evolve with the accelerator. Along with supporting additions and updates, future plans include further distribution of I/O processing and databases, and phasing out the DEC AlphaServer.

With the use of well tested and commercially available control system solutions, the ATLAS control system continues to be maintained by a very small staff.

REFERENCES

[1] Bollinger, L. M. et al., "ATLAS: A Proposal for a Precision Heavy Ion Accelerator at Argonne National Laboratory", Argonne National Laboratory, 1978, ANL-78-XX-68.

[2] F. Munson, D. Quock, B. Chapin, and J. Figueroa, "Argonne's ATLAS Control System Upgrade", International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS '99, Trieste, Italy, October 4-8, 1999.

[3] Vista Control Systems, Inc., Los Alamos, NM, USA.

[4] F. Munson, D. Quock, S. Dean, and K. Eder, "First Experiences Integrating PC Distributed I/O Into Argonne's ATLAS Control System", International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS 2001, San Jose, California, October 2001.

[5] Hytec Electronics LTD, Reading UK.

MIGRATION FROM OPC-DA TO OPC-UA

B. Farnham, R. Barillère, CERN, Geneva, Switzerland

Abstract

The OPC-DA specification of OPC has been a highly successful interoperability standard for process automation since 1996, allowing communications between any compliant components regardless of vendor. CERN has a reliance on OPC-DA Server implementations from various 3rd party vendors which provide a standard interface to their hardware. The OPC foundation finalized the OPC-UA specification and OPC-UA implementations are now starting to gather momentum. This paper gives a brief overview of the headline features of OPC-UA and a comparison with OPC-DA and outlines the necessity of migrating from OPC-DA and the motivation for migrating to OPC-UA. Feedback from research into the availability of tools and testing utilities will be presented and a practical overview of what will be required from a computing perspective in order to run OPC-UA clients and servers in the CERN network.

INTRODUCTION: OPC CURRENTLY AT CERN

At CERN, OPC is heavily used to control and monitor a large variety of devices. One commonly used technology stack allowing a human controller to control physical devices is a HMI and SCADA layer provided by Siemens [3] WinCC OA, WinCC OA communicates via its built-in OPC driver (an OPC Client) to an OPC server and the OPC Server communicates with the physical device in question.

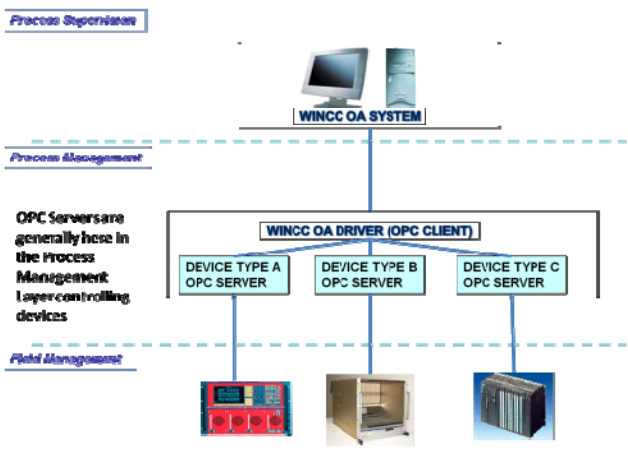


Figure 1: Common current usage of OPC at CERN as device orientated middleware.

The device vendor generally provides hardware plus an OPC Server capable of driving it. Current devices driven by OPC at CERN include:

- PLCs, providing low level control and monitoring of systems.
- High and low voltage power supplies, powering detector tubes and front end electronics.
- VME crates, providing specialist control of peripherals.
- Embedded Local Monitor Boards (ELMBs), a CERN proprietary I/O module for monitor/control of front end equipment.

Prior to OPC-UA, the OPC Foundation produced a series of sibling OPC specifications, amongst which was OPC-DA. These sibling specifications are hereafter referred to as OPC Classic. OPC-UA [1][2] is intended to be the modern successor to OPC Classic. CERN currently has only OPC Classic applications in production systems, however, there are some OPC-UA evaluation projects currently in progress.

OPC-UA HIGHLIGHTS

Platform Independence

OPC Classic is tied to Microsoft platforms by its dependence on COM/DCOM for security and inter-process communication. The OPC-UA specification has a built-in means of security, transmission and message encoding based on modern industry standards. Dropping the COM/DCOM dependency frees OPC-UA client and server implementations from the Microsoft platform.

Embedded Platforms

An important corollary of the point above is that OPC-UA servers can be written for embedded platforms – a device could have its own OPC-UA server built in. For scalability, the full functionality of the OPC-UA specification is chunked into discrete profiles describing subsets of the full feature set: An embedded OPC-UA server need only implement profiles relevant to its operating constraints.

Improved Security

Classic OPC has no intrinsic security; this is delegated to the COM/DCOM layer. OPC-UA has a comprehensive security model built in based on Public Key Infrastructure providing application identity and secure channel client/server communication. The specification also describes a means for exchanging user identity for application specific user authorization and authentication.

Improved Modelling

The OPC-UA standard provides an extensive vocabulary for modelling devices and processes, including being able to type components (allowing clients to establish semantic information) and to express relationships between components (allowing clients easier browsing between related components).

Message Transmission and Encoding Options

An OPC-UA slogan is 'from the device to the enterprise'. OPC-UA has been designed to allow

applications to encode and transmit messages in a manner most suited to their operating environment and constraints. There are two means of transmission: An efficient low level means, based on TCP/IP, called tcp.opc and ubiquitous HTTP. There are two means of encoding messages: UABinary, which is small on the wire and requires less processing overhead than the alternative XML/SOAP encoding which allows for the possibility of OPC-UA messages to be consumed by a wide variety of applications (i.e. not just OPC-UA). OPC-UA applications providing software interfaces to devices are expected to prioritise efficiency, whereas OPC-UA applications providing high level information publication are expected to prioritise support for the widest client base.

MOTIVATION IN MOVING FROM OPC CLASSIC TO OPC-UA

The primary motivation for migrating from OPC Classic is that its message transmission is based Microsoft's proprietary COM/DCOM. COM/DCOM has been de-emphasized in favour of .NET's Windows Communication Framework. OPC-UA on the other hand describes two non proprietary means of message transmission, tcp.opc or HTTP. OPC-UA applications can choose to support one or both transmission types.

Linux compatibility. Some administrators of Detector Control Systems (DCS) have selected Linux as their preferred operating system but are forced to maintain windows machines in order to host OPC Classic applications. Linux compatible OPC-UA replacements would allow administrators to remove these windows nodes thereby reducing maintenance complexity by homogenizing the host operating system, Linux, across the DCS. Devices supplied with embedded OPC-UA servers would further simplify DCS administration: The device hosts its own OPC-UA Server.

OPC-UA security is standards based and easier to configure. Restricting access to OPC Classic applications is possible but experience has shown that setting up DCOM security configuration is error prone and the results fragile to operating system updates and patches. OPC-UA application authenticity is provided by standard x509 certificates which every OPC-UA application is required to uniquely own. OPC-UA applications only accept session requests from other OPC-UA applications proffering trusted certificates. These same certificates are used to provide secure communications channels for client server sessions in which all messages can be signed (preventing message tampering) or signed and encrypted (additionally preventing eavesdropping).

OPC-UA allows for more intuitive server address spaces. As mentioned in OPC-UA highlights above, OPC-UA has some quite considerable changes and enhancements over OPC Classic. OPC Classic provides a simple address space in a tree structure of branch nodes containing branch nodes and leaf nodes. A common idiom in OPC Classic is to have write-only leaf nodes, which, when written to, invoke an action on the OPC server/device. To turn on/off a channel in an industrial power supply, for example, an OPC Classic user often writes true/false to a write only OPC item. The idiom works but to the uninitiated user it is not obvious that this is the standard means of turning a channel on. OPC-UA servers describe their address spaces in objects (as in Object Orientation) whereby objects contain fields, and methods which have an effect on those fields. An OPC-UA server for an industrial power supply could model a channel as an object with fields representing current, voltage etc. and a method, parameterised with a Boolean, to turn the channel on and off. This interface is more self-explanatory. Views are another useful OPC-UA address space enhancement. Similar to database views, OPC-UA views exist to provide subsets of the full address space tuned to a specific perspective. For example, a cooling and ventilating application is more likely to be interested in temperature sensor values of an industrial power supplies than, say, voltage and current read outs. An industrial power supply OPC-UA server could provide views tuned for such perspectives.

OPC-UA IMPLEMENTATION AVAILABILITY

Stacks

The OPC Foundation provides their corporate members with reference OPC-UA implementations, stacks, of the standard. Stacks are available in three languages: ANSI C, .Net and Java. The primary goal of the stacks is providing communications interoperability. At time of writing only the .Net stack supports both transmission means (opc.tcp and HTTP) and both message encodings (UABinary and XML/SOAP). The C stack and Java stack provide only opc.tcp transmission and support for binary encoded messages. In terms of the C stack this is not unreasonable, it is the most likely to be used for embedded or high performance environments where HTTP/SOAP would not be the natural choice. The .Net stack is the most fully featured but is limited to the .Net runtime, essentially limiting it to Microsoft platforms.

Toolkits

These are built on top of the stacks and aim to provide a much more complete SDK to aid and simplify application development (providing programmatic session management for example). The toolkits generally inherit any omissions or limitations in functionality from the stacks upon which they are built, the Java SDKs surveyed, for example, do not currently support HTTP transmission or XML/SOAP encoding.

There are various stack vendors available but the main players are:

- Unified Automation [5] - Have C, C++ and Java toolkits. Applications built with the Java toolkit are naturally cross platform, application hosts require JRE6. The C and C++ toolkits can, on request, be

cross compiled by the vendor to various Windows/Linux platforms including embedded variants and realtime operating systems.

- Softing [4] - Have a .Net (requiring the .Net 3.5 runtime) and C++ toolkit available. On request the vendor can compile the C++ toolkit for Windows or Linux.
- Prosys [6] - Have a Java toolkit, application hosts require JRE6.

Diagnostic Tools

An important tool for OPC deployment is a simple, visual OPC client, allowing verification and problem diagnosis of server installations. Unified Automation, provide UaExpert: A free, stable and sufficiently fully featured OPC-UA client available on Linux and Windows. For low level diagnosis, ascolab [8] provide a Wireshark [7] plugin allowing an engineer to view client/server messages (for unencrypted client/server sessions).

FUNCTIONALITY TESTS

The following tests were carried out using available implementations from various vendors. In all cases only UABinary message encoding was used and opc.tcp transmission.

Cross Stack and Cross Platform Interoperability

In every attempted permutation (see table below) of client/server toolkit and platform the connecting client was able to connect to, and browse, the server without problems:

Table 1: Permutations Tested for Cross Platform and Toolkit Client/Server Interoperability: All Succeeded

Client Description	Server Description	Server Description	Server Description
Java on Windows	Java on Windows	C++ on Windows	C++ on Linux
C++ on Windows	C++ on Windows	C++ on Linux	Java on Windows
C++ on Linux	C++ on Linux	C++ on Windows	Java on Windows
Java on Linux	C++ on Linux	C++ on Windows	Java on Windows

OPC-UA as Device Middleware

OPC Classic is heavily used at CERN as device middleware (see figure 1), OPC-UA's applicability for device middleware was tested. The test stack consisted of a WinCC OA HMI and SCADA layer, using its built-in OPC-UA driver to communicate with a custom made OPC-UA server providing control and monitoring of simulated hardware.

The WinCC OA system (version 3.10) and its OPC-UA driver ran on Windows 7. The OPC-UA server was built using Softing's C++ toolkit, the OPC-UA server and hardware simulation processes ran on Scientific Linux 5.

Table 2: Test Results of OPC-UA Functionality for Device Middleware

Functionality	Results/Notes
OPC Classic Functionality in OPC-UA	
Create a simple, browseable address space for device.	The OPC-UA server builds an address space to represent the device, browseable from WinCC OA.
WinCC OA datapoints could be mapped to server variables.	WinCC OA datapoints have a 'peripheral address' configuration, allowing mapping between them and OPC-UA server address space elements.
WinCC OA Datapoints updated as hardware values change.	WinCC OA defines subscriptions on the OPC-UA server, with publishing intervals. The OPC-UA server sends cyclic notifications to WinCC OA, updating the datapoints.
Can write values to server variables	WinCC OA's dpSet() command uses peripheral address mappings to send write commands to the OPC-UA Server, on receipt, the server writes hardware values.
OPC-UA Enhancements	
Server object methods can be browsed and invoked	The OPC-UA Server can provide methods on objects. These can be invoked using the UaExpert client. Siemens confirmed this is not available in WinCC OA version 3.10.
Server views can be browsed and datapoints mapped to view fields	As above, the OPC-UA server can provide views, browseable from UaExpert. Siemens confirmed this is not available in WinCC OA version 3.10.
OPC-UA Security	
OPC-UA applications only communicate with trusted applications	The OPC-UA Server only communicates with trusted clients however it was possible to initiate communications between WinCC OA and untrusted servers. The vendor is investigating this.
OPC-UA client/server traffic can be signed or signed and encrypted.	The OPC-UA Server supports this, tested in session with UaExpert, however attempts to create secure sessions between the server and WinCC OA failed. The vendor is investigating this.

Overall, OPC-UA analogs of OPC Classic functionality, essential for device control and monitoring, work in the surveyed OPC-UA implementations. Nice-to-have OPC-

UA device modelling enhancements were not present in the WinCC OA version used. WinCC OA security issues are being investigated by the vendor.

INFRASTRUCTURE REQUIREMENTS OF OPC-UA APPLICATIONS AT CERN

The main difference for system administrators is adapting to OPC-UA's security requirements.

Firewall Settings

OPC-UA has been designed to be firewall friendly. For tcp.opc traffic the administrator need only open the port required by each endpoint. Endpoints using HTTP transport require port 80 open.

Application Certificates

CERN already has infrastructure in place for generating and managing the type of signed certificates OPC-UA applications require: The CERN CA (Certificate Authority). The CERN CA is visible from both CERN's technical and general purpose networks. Currently, obtaining a CERN signed certificate requires non-trivial manual interaction with the CERN CA website. This effort would be required on each application installation and additional effort to renew certificates before they expire (CERN certificates generally expire after one year). Planned CERN CA enhancements include provision for programmatic certificate generation and renewal. Once in place, OPC-UA applications could be delivered with a module which communicates with the CERN CA interface to automate (as far as security policy allows) the process of obtaining signed certificates as part of the installation procedure and certificate renewal.

OPC AS HIGH LEVEL MIDDLEWARE

OPC-UA's features make it an interesting candidate for high-level middleware providing a means of inter-system communication and publication of enterprise level summary data (fig. 2).

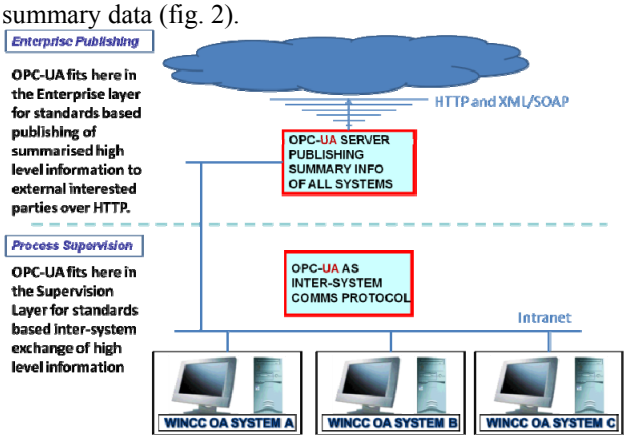


Figure 2: OPC-UA in high level middleware roles.

- OPC-UA is a common standard, compatible with a range of 'off the shelf' components.
- OPC-UA provides secure client/server sessions, with message integrity and confidentiality.
- opc.tcp transmission and UABinary encoding are designed for bandwidth and processing efficiency.
- HTTP transmission and XML/SOAP encoding opens the possibility of delivering messages to a broad base of non OCP-UA specific clients.
- Toolkit interoperability allows for creating a communications system between applications dispersed over heterogeneous platforms.
- Service discovery via OPC-UA discovery servers.

Whilst the features above are very promising facets of OPC-UA for high level middleware, the technology is, at time of writing, insufficiently mature for the purpose for three main reasons:

- During the evaluation, secure sessions failed to work between WinCC OA and an OPC-UA server. The vendor is investigating.
- HTTP transmission and XML/SOAP encoded messages are only currently available in the windows centric .Net tools.
- The final 'Discovery Server' section of the specification is currently in draft; however it is expected to be released this year.

CONCLUSION

OPC-UA is clearly a most compelling candidate for control and monitoring middleware between SCADA systems and device layers. OPC Classic, the incumbent CERN middleware, is based on deprecated COM/DCOM technology and OPC-UA is its successor, based on modern industry standards. The full functionality described in the specification was not found to be available using the current technology and tools surveyed. Despite this, however, current OPC-UA tooling would be sufficient (once security issues are resolved) to provide applications to match and exceed functionality currently provided by OPC Classic applications, most notably in that existing OPC-UA tools allow for interoperable cross platform implementations. Features described in the OPC-UA specification make it an interesting candidate technology for standards based, inter-system communications. However, implementations of some of these features (e.g. server discovery) were absent in the tools surveyed. The evolving OPC UA toolset will be monitored for implementations of these features and verified to ensure they fully cover these expectations.

REFERENCES

[1] The OPC-UA Specification. www.opcfoundation.org
[2] W. Mahnke, S. Leiner, M. Damm, 'OPC Unified Architecture', Springer 2009.
[3] Siemens Automation. www.automation.siemens.com
[4] Softing. www.softing.com
[5] Unified Automation. www.unified-automation.com
[6] Prosys. www.prosysopc.com
[7] Wireshark. www.wireshark.org
[8] ascolab. www.ascolab.com
[9] EN/ICE OPC Support website: www.cern.ch/wikis/display/EN/OPC+Support

J-PARC CONTROL TOWARD FUTURE RELIABLE OPERATION

N. Kamikubota#, N. Yamamoto, H. Nakagawa, S. Yamada, K.C. Sato, T. Katoh, J. Odagiri
KEK / J-PARC, Tokai-mura, Ibaraki, Japan
N. Kikuzawa, Y. Kato, M. Kawase, Y. Ito, T. Suzuki, T. Ishiyama, S. Fukuta, H. Yoshikawa, H. Sakaki, H. Sako, H. Takahashi
JAEA / J-PARC, Tokai-mura, Ibaraki, Japan
S. Yoshida, M. Takagi, S. Motohashi, T. Iitsuka
Kanto Information Service (KIS), Tsuchiura, Ibaraki, Japan
H. Nemoto, ACMOS Inc., Tokai-mura, Irbaraki, Japan
D. Takahashi, Mitsubishi Electric System & Service Co. Ltd., Tsukuba, Ibaraki, Japan

Abstract

The J-PARC accelerator complex comprises proton LINAC, 3-GeV RCS, and 30-GeV MR. The J-PARC is a joint project between JAEA and KEK. Two control systems, one for LINAC and RCS and another for MR, were developed by two institutes. Both control systems use the EPICS toolkit, thus, inter-operation between two systems is possible. After the first beam in 2006, beam commissioning and operation have been successful.

The current status of J-PARC control structure is given. Components are introduced and reviewed with reliability issues: A plan to develop new GUI applications for operators is given.

INTRODUCTION

A high-intensity proton accelerator complex, J-PARC (Japan Proton Accelerator Research Complex), consists of three accelerators: a) 180-MeV LINAC (energy upgrade to 400 MeV is underway), b) 3-GeV RCS (Rapid Cycling Synchrotron), and 30-GeV MR (Main Ring). J-PARC is a joint project between two institutes: JAEA in Tokai-mura and KEK in Tsukuba.

The construction of J-PARC facilities started in 2002 in Toka-mura. The first proton beam to LINAC was in 2006, followed by RCS (MR) in 2007 (2008), respectively. In March, 2011, we have achieved stable beam operation at the power of 220 kW for neutron productions, and of 140 kW for neutrino productions.

In the very early phase of the project, a decision was made that JAEA is in charge of LINAC and RCS, and KEK is of MR. Discussions on J-PARC controls were also made around 2001. As a result, we decided to have two independent control groups. However we had some agreements: a) both use the EPICS toolkit, b) three accelerators must be controlled from a single control room, c) have single personnel protection system (PPS), and so on. Even after the beam operation started, we still keep two control groups.

The above history has made our control structure very difficult to understand. In the past, general design views of J-PARC controls were reported in [1,2]. System and infrastructure of JAEA was given in [3]. Control overview was presented in [4]. In this report, our control structure is reviewed, with discussions on reliability issues.

#norihiko.kamikubota@kek.jp

CONTROL SYSTEMS OVERVIEW

Components Overview

A brief summary of control components is given in Fig. 1. It is apparent that JAEA (LINAC and RCS) and KEK (MR) have different policies.

Accelerator	OPI Application Basic	OPI Application High-Lvl.	IOC(VME) OS,H/W	Drivers
Linac	Java +MEDM	XAL /JCE	VxWorks PowerPC	- VME I/O Modules - TeraDev for PLC
RCS	Java +MEDM	SAD	VxWorks PowerPC	- VME I/O Modules - TeraDev for PLC
MR	MEDM or EDM	SAD Python ROOT	Linux Intel-based (+PLC/F3RP 61-Linux)	Network devices -NetDev for PLC -WE7000 drivers

Figure 1: Overview of control components at J-PARC.

OPI (GUI) Applications

JAEA has developed stand-alone Java applications for basic GUI applications. In the early phase, JCA libraries, which handle Java to EPICS connections, had small bug. JAEA modified the library to avoid problems [5]. For high-level applications used in beam commissioning studies, XAL was introduced from SNS [6]. Later JCE, simulating SAD script using Java [5], was developed and used by JAEA[7].

KEK introduced MEDM and EDM as basic GUI tools. SAD script, Python script, and ROOT, were used for high-level applications.

The key idea of JAEA is that Java should be the primary platform for all applications. In addition, JAEA developed tools by themselves, except for XAL. However, rack of rapid development tool allowed some MEDM applications in LINAC and RCS.

In turn, KEK introduced EPICS standard tools, based on experiences of KEKB controls. This selection saved start-up man-powers, and enabled rapid developments of GUI applications for MR. Up to now, number of applications registered to the MR launcher is about 270.

IO Controller and Signal Front-end

Both groups selected VME-bus computer as a default IO controller. JAEA selected a Power-PC based CPU board with real-time operating system (vxWorks). There are about 120 (30) pieces of CPU boards for LINAC (RCS), respectively. Standard IO boards of VME-bus are used to handle signal cables. In addition, pieces of Yokogawa PLC (Programmable Logic Controller) with ladder logic were introduced as local controllers of various power-supplies.

In KEK, an Intel-based CPU board and Linux operating system were selected. They are used as disk-less Linux computers. KEK enhanced to use network-based devices, such as Yokogawa PLC and WE7000 measuring station [8,9]. In 2011, we have about 90 pieces of CPU boards for MR, however, 60 of 90 CPU boards do not have any IO boards (see Fig. 2).

In 2008, a new CPU module for Yokogawa PLC (F3RP61-2L) appeared. It can be a Linux-based IO controller with PLC IO modules [10]. In 2011, there are about 30 pieces in MR.

Selections by JAEA were traditionally safe for 25Hz machines. This is because decision was made around 2002. Before the decision by KEK in 2007, we had much discussion whether we chose the same operating system (vxWorks) as JAEA or not. Finally KEK selected Linux. Use of Linux allows easy maintenance and customization of control software. It is worth noting that MR is a slow-cycle machine (a few seconds), thus, real-time features of vxWorks are not needed.

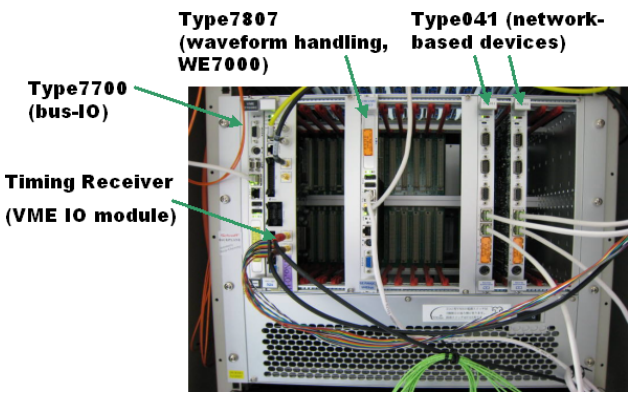


Figure 2: Typical IO Controllers for MR.

Network System

Network system for J-PARC accelerators was introduced around 2004 by JAEA. The system has intelligent core loops to ensure high redundancy [3]. Later in 2006, KEK joined to use the system together. In 2011, we have 80 edge switches for LINAC, 24 for RCS, and 5 for MR. The bandwidth between buildings has been 1 Gbps, partly upgraded to 10Gbps in the summer, 2011.

The network system is logically divided into sub-networks (vlan), which correspond to accelerators (LINAC, RCS, MR) and other facilities. Thus, most of network traffics are localized within single sub-network. For EPICS protocol, we put a few gateways (CA gateway) to enable communication between sub-networks.

Basic services of TCP/IP networks (DNS, NTP) are managed by JAEA, while user authentication service (LDAP) is by KEK. A web server has been operated by KEK to indicate accelerator status and information.

Computers and Storage Disks

JAEA has used rack-mount servers for disk storage. In 2011, total amount is around 16 TB. Standard desktop PCs, running RHEL, have been used as operator's consoles as well as platforms to run control applications.

KEK started to use a dedicated disk storage system and blade-type servers in 2007. In 2011, total amount of disks is around 20 TB, and more than 20 pieces of blade-type servers, running Scientific Linux, are used in operation. Control applications run on blade-type servers, while GUI screens are displayed on thin client terminals [11].

RDB (Relational Database) and Data Archive

Both JAEA and KEK have been using PostgreSQL RDB engines. JAEA has used RDB features for: a) automatic generation of configuration files for EPICS IO Controllers, and b) handling operation parameters (for example, timing-signal settings [12]). In addition, data archive system has been developed using RDB [13].

KEK has used RDB for backup purposes. Databases for a) IP addresses and EPICS record names in MR, b) copies of GUI screens on operator consoles, have been managed using RDB. One can search them through Web-based interfaces. For data archive, channel archiver, a standard tool of EPICS, was introduced and has been used since the beginning of MR beam commissioning [14].

ISSUE FOR RELIABLE OPERATION

Control Room

In J-PARC, all accelerators (LI, RCS, and MR) are controlled from a single control room. We have arranged console desks to form a rectangular shape, in order to force people stay inside (Fig. 3). Two workspaces and one meeting desk are nearby. This structure encourages tight communication between operators, a shit leader, and commissioning staff. In addition, people from experimental facilities (MLF, HD, NU in Fig. 2) can stay in the control room for communication. We believe that this idea is essential to improve reliability.

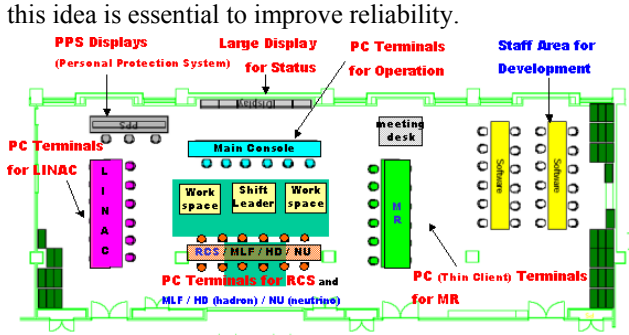


Figure 3: Schematic view of control room for J-PARC.

Safety System (PPS)

The safety is fundamental for high power proton machines like J-PARC. The PPS is to protect personnel from the radiation and other hazards caused by accelerator operations. In order to avoid confusions originated from different policies of JAEA and KEK, we assigned a single person for design and construction for whole of the PPS.

The PPS must be highly reliable and fail-safe. The system was developed using PLC with ladder logic, but no usual computer with operating system. Moreover, we asked two companies to develop PLC ladder logic independently. Thus, having two PLC-based sequences makes the PPS system extremely redundant. More detailed description of PPS is given in [15].

United Sub-Systems

Some sub-systems were introduced for LINAC and RCS by JAEA first, later KEK disagree to extend it to MR, but developed a different system with a link to the JAEA system. The examples of "united" sub-systems are: a) MPS (Machine Protection System) and b) timing system.

The MPS for LINAC and RCS was developed by JAEA [3]. The system has a simple tree-type topology. Because the MPS system for MR was requested to have links to downstream experimental facilities, more complicated logic scheme was needed. As a result, KEK developed a new MPS using FPGA-based logic controllers [16].

The studies of timing system for J-PARC were made in very early phase. NIM modules for signal distribution to all buildings and VME-bus timing IO modules were developed in 2002-2003 by JAEA [3,17]. However, software strategy by JAEA was designed only for 25Hz machines. As a result, KEK decided to use the same hardware (NIM and VME modules) to link to the JAEA timing system, but developed another software framework to fit slow machine cycle of MR (a few seconds) [18].

Electric Logbook

An electric logbook, Zlog, was developed using RDB and Python script by KEKB [19]. We decided to use Zlog for all J-PARC accelerators since early phase of its operation. Operators are responsible to insert machine events. One can watch at them remotely by a Web interface.

Since Zlog is RDB-based, it has advanced features as: a) search entries to search for past events, b) automatic generation of machine study report, c) recalling a shift summary, and so on. In addition, automatic insertion mechanism of machine faults has been developed. It has been in evaluation in operation.

PLAN FOR NEW GUI FOR OPERATORS

Up to now, we already experienced three-year beam operation for experimental facilities. However, operation experience shows that two control systems often make operators distressed. For example, different GUI look-and-feels, separated MPS screens, independent data archive systems, and so on. In addition, we do not have an alarm system, which informs pre-critical events of devices before a severe fault. Moreover, many of GUI applications for MR were developed for accelerator experts, not for operators. A miss-operation by operator would cause a serious machine stop.

Considering demands of further power upgrade and longer beam delivery times in the future, we need a new GUI system for operators. We, two control groups, have started to discuss to develop common GUI screens of status and alarms, and an interface to connect to both data archive systems. Recently we have interested in CSS (Control-System Studio) [20], which has been widely introduced in EPICS-based control systems. An idea is that we add a CSS layer over existing control systems. As shown in Fig. 4, the CSS layer will contain: a) new GUI applications common for all three accelerators, b) a single alarm system for all three accelerators, and c) an archive viewer for two different archive systems.

In the summer, 2011, we succeeded to introduce CSS package with helps of CSS experts [21]. Three components of CSS (GUI, Alarm, Archive viewer) worked well. Evaluation and development will start soon.

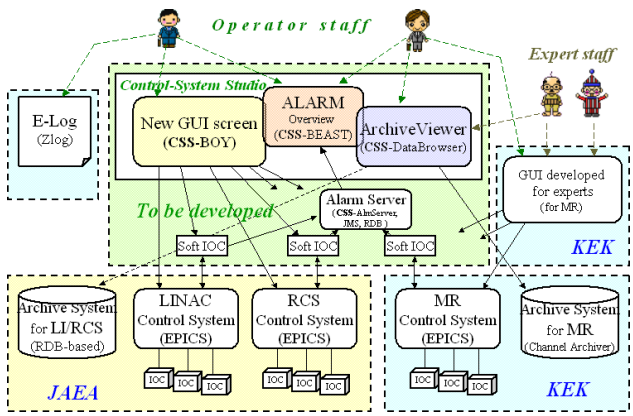


Figure 4: Relations of CSS layer and control systems.

ACKNOWLEDGEMENT

The components of control systems for J-PARC have been highly relied on fruitful products of EPICS toolkit. In addition, we thank EPICS collaborators for various discussions during construction and development. Thanks are also to operators and support staff members who have concern accelerator operation and system maintenance.

REFERENCES

[1] J.Chiba et al, "A Control System of the JHF Accelerator Complex", ICALEPCS 1999, Trieste, p.1-5
T.Katoh et al, "Present Status of the J-PARC Control System", ICALEPCS 2003, Gyeongju, Korea, p.1-5
[2] T.Katoh et al, "Present Status of the J-PARC Control System", PAC 2005, Knoxville, p.302-304

T.Katoh et al, "Towards the Commissioning of J-PARC", ICALEPCS 2005, Geneva, MO3.5-10

[3] H.Sakaki et al, "The Control System for J-PARC", APAC 2004, Gyeongju, Korea
H.Takahashi et al, "Control System of 3 GeV Rapid Cycling Synchrotron at J-PARC", PAC 2005, Knoxville, FPAT047
H.Yoshikawa et al, "Current Status of the Control System for J-PARC Accelerator Complex", ICALEPCS 2007, Knoxville, p.62-64

[4] N.Kamikubota, "J-PARC Status", presentation at EPICS collaboration meeting in Shanghai, March 2008

[5] H.Sako et al, "Beam Commissioning Software and Database for J-PARC LINAC", ICALEPCS 2007, Knoxville, p.698-700
H.Sako, "JCAL - A Java Channel Access Client Library", presentation at EPICS collaboration meeting in Kobe, October 2009

[6] C.K.Allen et al, "XAL Online Model Enhancements for J-PARC Commissioning and Operation", ICALEPCS 2007, Knoxville, p.494-496
T.Pelaia II et al, "XAL Status", ICALEPCS 2007, Knoxville, p.34-36

[7] RCS commissioning team selected SAD instead of XAL/JCE.

[8] M.Takagi et al, "Network-based Waveform Monitor for J-PARC Accelerator Complex", ICALEPCS 2003, Gyeongju, Korea, p.497-499
N.Kamikubota, "Applications of Network-based Controllers at KEK", presentation at EPICS seminar in Shanghai, April 2005

[9] J.Odagiri et al, "EPICS Device/Driver Support Modules for Network-based Intelligent Controllers", ICALEPCS 2003, Gyeongju, Korea, p.494-496

[10] A.Uchiyama et al, "Development of Embedded EPICS on F3RP61-2L", PCaPAC 2008, Ljubljana, p.145-147
J.Odagiri et al, "Application of EPICS on F3RP61 to Accelerator Control", ICALEPCS 2009, Kobe, p.916-918
N.Kamikubota, "Demonstration of Embedded EPICS on F3RP61 PLC", presentation at EPICS collaboration meeting in Taiwan, June 2011

[11] S.Yoshida et al, "Console System using Thin Client for the J-PARC Accelerators", ICALEPCS 2007, Knoxville, p.383-385

[12] H.Sako et al, "Relational databse System for J-PARC LINAC and RCS", ICALEPCS 2005, Geneva, PO1.080-7
H Takahashi et al, "Database for Control System of J-PARC 3GeV RCS", ICALEPCS 2007, Knoxville, p.567-569
H Takahashi et al, "Timing Delay Management database for J-PARC LINAC and RCS", ICALEPCS 2009, Kobe, p.450-452

[13] H.Takahashi et al, "Data Acquisition System for J-PARC 3GeV RCS", ICALEPCS 2005, Geneva, PO1.009-1

[14] N.Kamikubota et al, "Data Archive System for J-PARC Main Ring", IPAC 10, Kyoto, p2680-2682.

[15] Y.Takeuchi, "Personal Protection System of Japan Proton Accelerator Research Complex", ICALEPCS 2003, Gyeongju, Korea, p.404-406

[16] H Nakagawa et al, "The Accelerator protection System Based on Embedded EPICS for J-PARC", ICALEPCS 2009, Kobe, p.406-408

[17] F.Tamura et al, "J-PARC Timing System", ICALEPCS 2003, Gyeongju, Korea, p.247-249

[18] N.Kamikubota et al, "Overview of Timing System for-PARC MR", PASJ meeting, Higashi-hiroshima, Japan, 2008, p.286-288 (in Japanese)

[19] K.Yoshii et al, "Web-based Electric Operation Log System - Zlog system", ICALEPCS 2007, Knoxville, p.299-301

[20] J.Hajie et al, "Control System Studio (CSS)", ICALEPCS 2007, Knoxville, p.37-39
J.D.Pacel et al, "CSS - We didn't Invite it, We Made it better", ICALEPCS 2009, Kobe, p.114-116

[21] Kay Kasemir, ORNL, and K.Furukawa, KEK

FAST BEAM CURRENT TRANSFORMER SOFTWARE FOR THE CERN INJECTOR COMPLEX

M. Andersen*, L. Jensen. CERN, Geneva, Switzerland

ABSTRACT

The fast transfer-line Beam Current Transformers (BCTs) in the CERN injector complex are undergoing a complete consolidation to eradicate obsolete, maintenance intensive hardware. The corresponding low-level software has been designed to minimise the effect of identified error sources while allowing remote diagnostics and calibration facilities. This paper will present the front-end and expert application software with the results obtained.

INTRODUCTION

The Transformer Integrator Card (TRIC) a new digital acquisition module, was designed to replace obsolete analogue integrators used to acquire the beam intensity from the Fast Beam Current Transformers (FBCT) in the CERN Proton Synchrotron Booster (PSB) and CERN Proton Synchrotron (PS) transfer lines.

The older Fast BCT acquisition electronics was, for the most part, analogue and provided poor facilities for remote adjustment and diagnostics. The output of the older FBCT system was essentially limited to a single value corresponding to a beam's total intensity. While the sample-to-sample reproducibility was within the 1% required for normal operation, the absolute errors could be considerable due to time-variations in the analogue electronics. Maintaining this system often required cumbersome expert interventions at the site of the installed hardware. This approach is outlined in Figure 1.

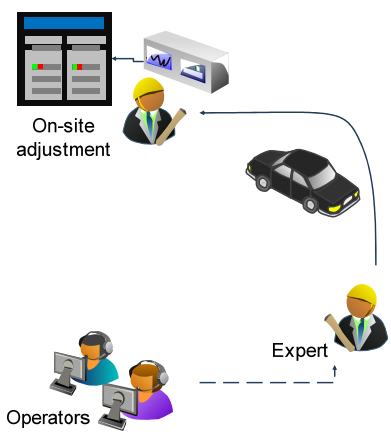


Figure 1: Interaction with old system.

The following outlines the main drawbacks of the old system:

- No ability to precisely time the acquisition to the beam signal, leading to the integration of additional noise in the measurement
- No ability to perform remote calibration or any remote diagnostics
- No possibility to discern and measure multiple isolated beam bunches and beams of multiple injections
- Parameters such as measurement gains had to be changed on-site via analogue potentiometers using a screwdriver
- The DC measurement offset had to be subtracted by the software. This made it impossible to compare the results from two distinct FBCT's in hardware and prevents the implementation of a hardware-only Beam Loss Watchdog. Reliability is therefore reduced due to this dependence on software and its associated complexity.

The new system was designed to address the above issues while maintaining or improving the satisfactory relative measurement accuracy.

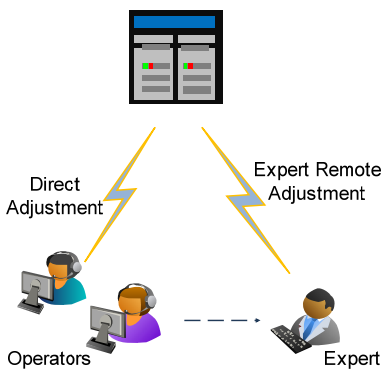


Figure 2: Interaction logistics for the new system.

TRIC HARDWARE SOLUTION

The TRIC is a VME-based acquisition module equipped with two analogue inputs each sampled at 200 MHz and subsequently processed by a large FPGA [1]. The TRIC card incorporates all the required functional elements for making reliable beam intensity acquisitions. The flexible and modular nature of the FPGA digital platform allows multiple data integration schemes which are explained below:

Parallel integrator – uses single window beam capture scheme with an accompanying offset capture. This integration is the most commonly used type for ordinary PS and PSB transfer line measurements. The

DC offset measurement can be specified to be performed at any time interval when there is no beam in the line.

Calibration integrator – single calibration window with accompanying DC offset capture as in the Parallel integrator above.

Bunch integrator – allows up to 32 possibly overlapping measurement gates to be defined and acquired using a single trigger signal. Each measurement has its own delay and capture length.

Multigate integrator – stores its values as a set of 1024 back-to-back capture gates with the same programmable gate duration. This mode provides an "oscilloscope" type overview and is extensively used in setting up timing parameters for the "single shot" integrators described above [2].

SOFTWARE ARCHITECHTURE

The software architecture for the new Fast BCT acquisition system is displayed in Figure 3. It consists of:

1. *BCTFPS*, a FESA acquisition server running locally on the Linux or LynxOS VME front end computers.

2. *BCTFPS_Monitor*, a Java based expert GUI application used for adjusting measurement parameters and validating acquired results.

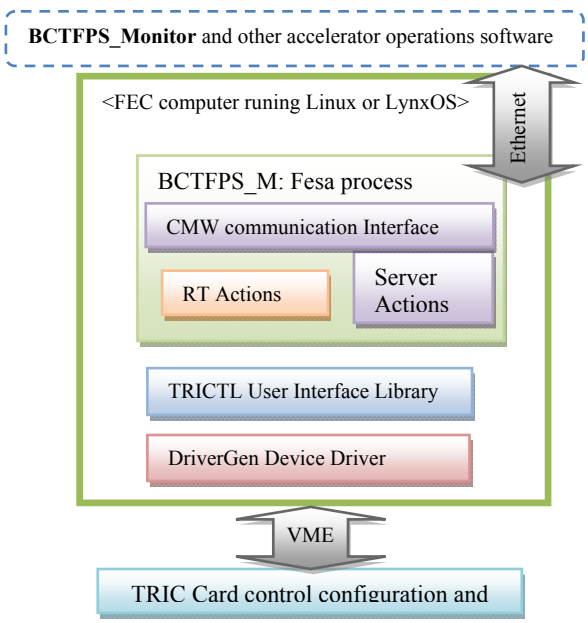


Figure 3: Software structure.

The *BCTFPS* is based on the FESA framework [3] allowing it to benefit from features such as real-time scheduling, beam-cycle synchronisation, user-notifications and use of a shared memory model.

Communication with the TRIC card is provided by a *TRICTL User Interface Library* where the register access to the TRIC hardware is made and data is decoded and pre-processed [4]. The low level hardware access is implemented using CERN's new standardized DriverGen driver platform.

The real-time part of the BCTFPS acquisition server consists of Prepare and *Acquire* actions triggered by PS or PS Booster central timing events linked to active elementary cycles. The following table summarises various steps taken during real-time actions.

Table 1: Functions Performed in Corresponding Real-Time Part of the BCTFPS Code

Real time SW Action	*Main functionality*
Prepare	✓ Reset integrators ✓ Configure each of the hardware integrators for the new cycle ✓ Select calibration scheme ✓ Setup cycle specific calibration parameters
Acquire	✓ Read timestamps, beam destinations and measurement status ✓ Read and store intensity measurements ✓ Scale *multigate* integrator values ✓ Compute intensity from raw data for verification ✓ Compute long term current statistics for the beam

INSTRUMENTATION EXPERT SOFTWARE

The *BCTFPS_Monitor* application (Fig. 4) is built for the adjustment of the timing as well as electrical and logical parameters for the low level *BCTFPS* acquisition server. It is coded using *Expert GUI* framework Java libraries developed by the CERN Beam Instrumentation Software Section [5]. The settings that can be applied are grouped into those applied for specific acceleration cycles and those that are common for all cycles. The integration gate length and its delay from the start trigger are, for example, settings which vary greatly from one accelerator cycle to another.

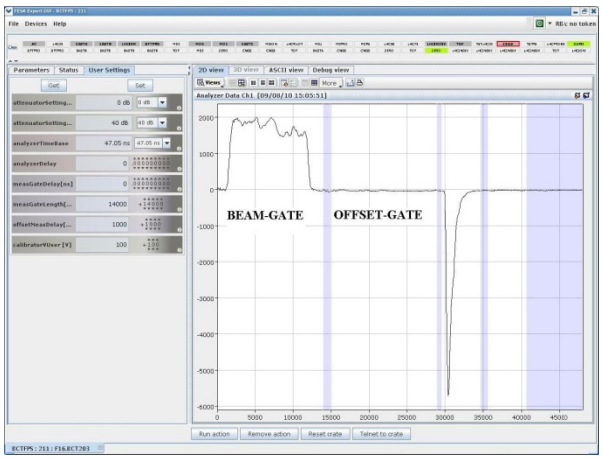


Figure 4: BCTFPS_Monitor GUI screenshot showing the beam intensity and negative calibration pulse with respect to the various integration gates.

The main part of the GUI is a large graph showing acquisition data from the 1024 data points of the *multi-gate* integration. The graph refreshes with every beam pulse passing in the transfer line. The beam acquisition and calibration pulse are superimposed on the acquisition gates using background colour cursors. The time base for the multigate integrator can be varied, achieving a 10ns resolution at its minimum setting. Figure 5 shows some of the shortest beam pulses extracted. The individual bunches of the *SFTPRO* beam cycle, ejected from the PSB, are <200ns long and spaced by ~300ns, and can easily be resolved using the smallest multigate integration time window.

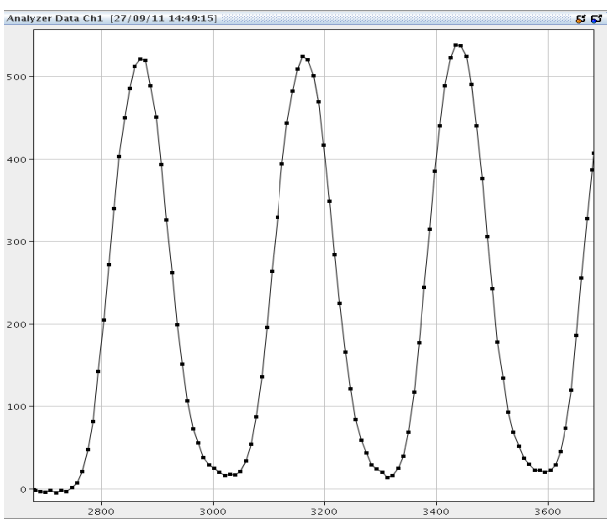


Figure 5: PBS ejected *SFTPRO* beam with *multigate* integration time increment of 10ns.

Increasing the increment acquisition time to tens, and even hundreds of ns, allows the expert to see the full picture for beams of longer duration. Figure 6 displays the same *SFTPRO* cycle at ejection from the PS, with the full 10μs beam pulse now visible.

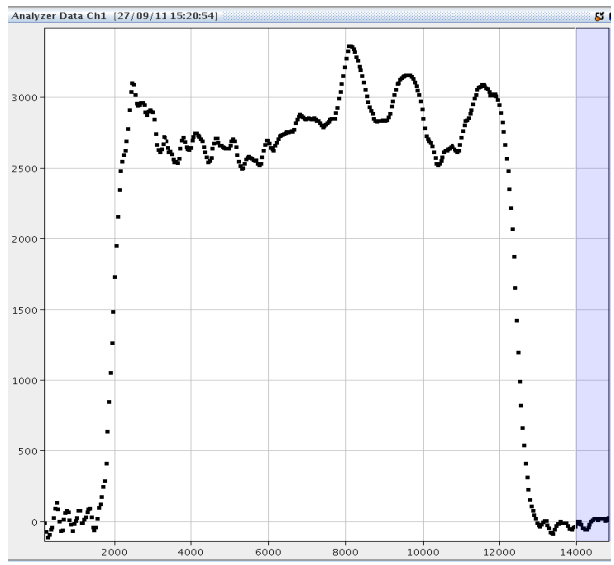


Figure 6: PS ejected *SFTPRO* beam with multigate integration time increment of 38ns.

MEASUREMENT REPRODUCIBILITY

The quality of the measurement data acquired by the new BCTFPS server was inspected with respect to different acquisition scenarios and compared to the old Fast BCT acquisition system. The first parameter studied was the relative error of the new system in the PSB transfer line compared to the intensity measured in the ring itself.

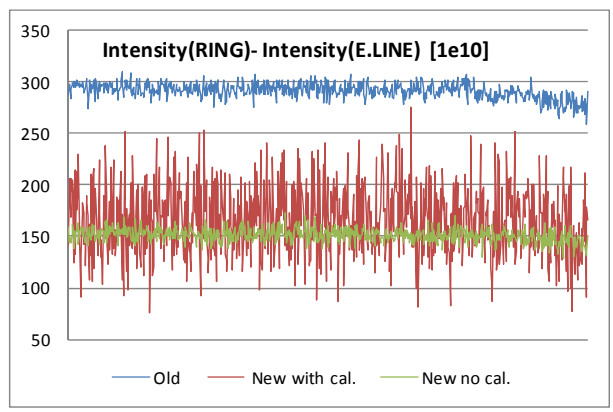


Figure 7: Comparison of the reproducibility for the old and new system. Using shot by shot calibration increases the noise on the measurement.

For the measurement in Figure 7, the average beam intensity was around 3e14 charges. The absolute value of the error is not important for the variation analysis, since the acquisition systems are not cross calibrated. It was seen that when the calibration was applied on a cycle by cycle basis additional noise was added to the final intensity result. This was found to come from the non-reproducibility of the calibration pulse. It was thus decided that rather than perform the calibration for each new measurement it was better to use a single

constant calibration factor. This factor was to be determined and applied by the expert in the instrument setup phase by taking the mean value from many calibration pulses. Once the calibration factor was fixed the remaining errors in the intensity measurement came from the front-end electronics and reproducibility of the beam injection process. The drawback of the "fixed calibration factor" scheme is that the system is not 100% autonomous and maintenance free.

Unique calibration factors have to be maintained for each of the two acquisition channels and each of the three attenuation levels (0dB, 20dB and 40dB). To set-up these factors, the expert uses a dedicated feature of the *BCTFPS_Monitor* GUI (Figure 8). The steps are the following:

1. Enable active calibration routine after every beam measurement
2. Unselect the "Freeze" radio-button in current attenuation level.

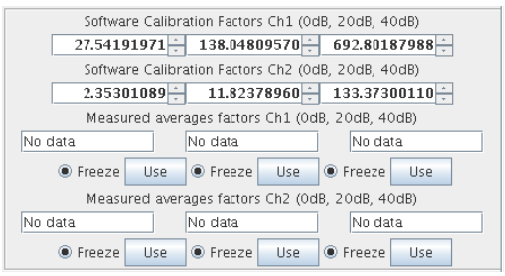


Figure 8: GUI panel for the calibration factor setup.

The system then starts acquiring calibration factors computed from the applied physical calibration pulses. The running sum of the 16 last measurements is displayed in the field.

3. Once the average is accumulated the Expert presses the "Use" button to subsequently set this value in a firmware register.
4. Disable active calibration and start using the new fixed calibration factor.

The quantitative improvement in the measurement reproducibility (i.e. less shot-to-shot variation) by using a "fixed" calibration factor has been significant. The following table displays the relative error of the total measured intensity for two distinct Fast BCTs [2].

Table 2: Reproducibility Error of old and new Fast BCT Acquisition Systems

Measurement System	BCT.BTTRA	BTY.TRA325
Original System	0.63%	0.23%
New system with Active Calib.	0.65%	1.2%
New system Const. Calib. factor	0.08%	0.22%

The last column showing BTY.TRA325 corresponds to the graph in Figure 8 above and shows how the relative pulse-to-pulse error was initially dominated in the new system by the shot-to-shot reproducibility of the calibration, returning to that of the old system once a constant calibration factor was applied. The downside of this solution is the fact that the new TRIC based system did not become totally maintenance free, as originally envisioned, since a periodic adjustment of the fixed calibration factor is now needed to correct for long term drifts in the acquisition electronics.

CONCLUSION

The new Fast BCT acquisition system has been significantly improved by new hardware and software. A highly modular digital integration module allows experts to cover many possible measurement scenarios by means of multiple integrator implementations keeping configuration settings limited and uncluttered. The additions of the on-board calibration circuit, as well as 1024 data point acquisition graph view, have greatly simplified instrument expert setup and maintenance procedures. The software solutions for both the front-end FESA server and the Expert GUI enable a seamless acquisition service to accelerator operations and a convenient parameter adjustment interface for the experts.

ACKNOWLEDGMENTS

I would like to acknowledge the invaluable work of A. Monera, D.Belohrad, L. Jensen, J-L.S.Alvarez, S.B.Pedersen and other colleagues at CERN-BE-BI/OP groups in preparing this paper.

REFERENCES

[1] A. Monera, D. Belohrad and G. Kasparowicz, *"Tric-2" TRansformer Integrator Card.* Geneva : CERN, 2010.

[2] A. Monera, *Upgrade of the CERN PSB/CPS Fast Intensity Measurements.* (Switzerland) Geneva : CERN, 2011.

[3] J. Paulluel, *Fesa Essentials.* Geneva : CERN, 2010.

[4] M. Andersen, *BI Front End Software functional specifications for the PSB/PS type Trransfer-Line BCT System [BCTFPS].* Geneva : CERN (EDMS), 2010.

[5] S. Pedersen, S. Bozyigit and S. Jackson, *Java Expert Gui Framework for CERN's beam instrumentation systems.* BE-BI, CERN. Geneva : CERN, 2011.

CSNS TIMING SYSTEM PROTOTYPE

G.L. Xu, G. Lei, Y.L. Zhang, P. Zhu, L. Wang, IHEP, Beijing, China

Abstract

Timing system is important part of CSNS. Timing system prototype developments are based on the Event System 230 series. We use two debug platforms, one is EPICS base 3.14.8. IOC uses the MVME5100, running vxworks5.5 version; the other is EPICS base 3.13, using vxworks5.4 version. Prototype work included driver debugging, EVG/EVR-230 experimental new features, such as CML output signals using high-frequency step size of the signal cycle delay, the use of interlocking modules, CML, and TTL's Output to achieve interconnection function, data transmission functions. Finally, we programed the database with the new features and in order to achieve OPI.

INTRODUCTION

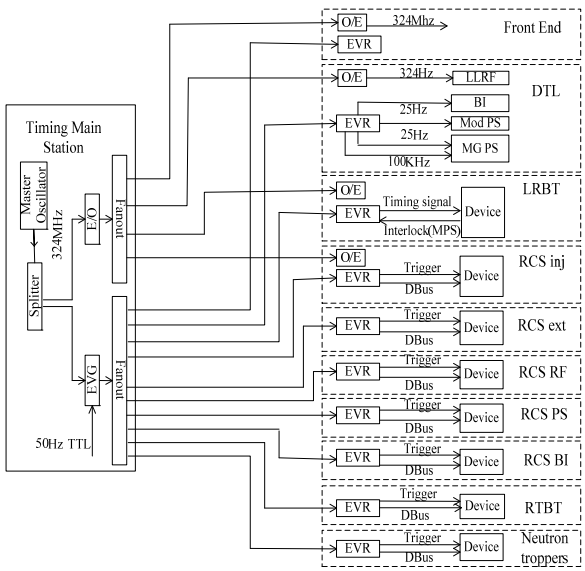


Figure 1: Timing System Outline.

The timing system using an Event system, that the event generator (EVG) and event receivers (EVR) as the main structures of a multi-level star structure(see Fig. 1), the core is EVG. The basic principle is, EVG distributed generation 8-bit parallel bus clock signal and 8-bit event codes, with 4 of 20 after a check code and / strings encoded as a stream of events, through the electro-optical conversion, optical fan-out module by and fiber (or cable) will be sent to a series of event streams EVR.EVR to receive the flow from the EVG's event to restore the 8-bit distributed bus clock signal and the 8-bit event code, and phase-locked clock generation and clock events EVG, EVR as a variety of operating reference clock. EVR can specify the port output clock signal; in the default event code EVR arrives, you can also specify the port in the pulse trigger signal output, and can produce a soft interrupt, the event code and time stamp stored in the FIFO memory with the memory can be read via the VME bus.

In BEPCII, event system has been successfully used to build a timing system, and long-term stable operation. CSNS timing system also intend to continue to use the event system[1]. Since 2004, the event system has also been upgrading , added a lot of features, the prototype of these new features was attempted. The CSNS timing system prototype provide the necessary basis for the construction of CSNS.

PLATFORMS AND DRIVER DEBUGGING

The software and hardware platforms of this experiment is as follows:

- Solaris2.8 of UNIX, mvme5100 CPU board, EPICS 3.13.8, Tornado2.1, VxWorks 5.4.2 .
- Red Hat Enterprise Linux 4, Kernel 2.6.9.Tornado 2.2.1, VxWorks 5.5.1, mvme5100 CPU board, EPICS BASE 3.14.8.2.

Modify some files of the driver，such as the routine ErCMLPat in the file drvMrfEr.c. Modify the souce code and the Db file.

HIGH-FREQUENCY STEP CYCLE DELAY

The control of the trigger delay is based on the RF cycle step (event clock = RF cycle * divide factor, of which divide factor can be 4, 5, 6, 8, 10, 12). As usual, the signal delay will be 12ns (see Fig. 2).This prototype utilize the CML output character of EVR230, of which up to 16 20-bit pattern configure register achieve the trigger delay control based on RF cycle step[2]. In fact, the maximum accuracy can be 1/(81MHz * 20)=617ps (see Fig. 3).

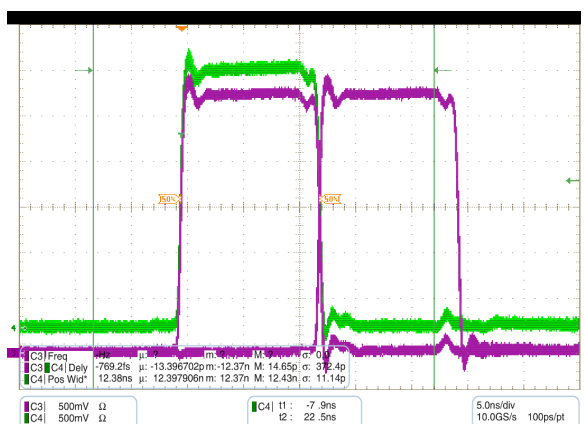


Figure 2: Delay is 12ns use common output.

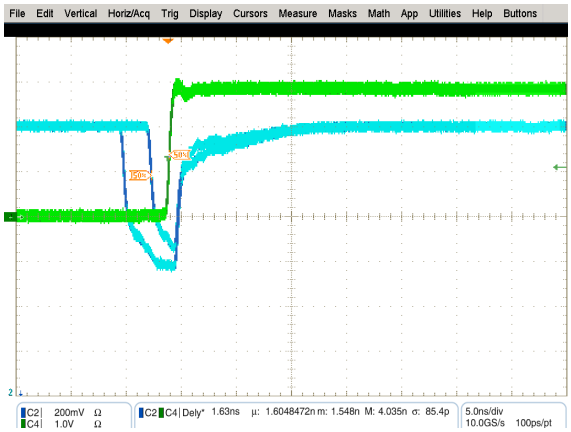


Figure 3: Delay is 600ps use CML output.

INTERLOCK FUNCTION

The event system of 230 series has interlock function. This function controls system can be used to participate and achieve with the machine protection system (MPS) of the interface. Using interface module UNIV-TTLIN, to receive electrical signals from the machine protection system, when the change occurs, the interlock signal level will become high level (see Fig. 5) from low level (see Fig. 4), the yellow signal will be closed .

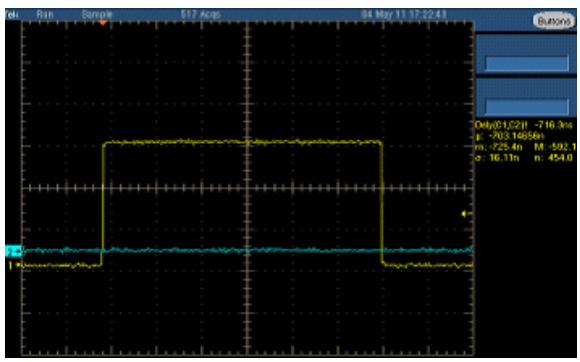


Figure 4: The blue signal is the input signal of interlock, its level is low when there are no interlocks. The yellow signal is the output of EVR.

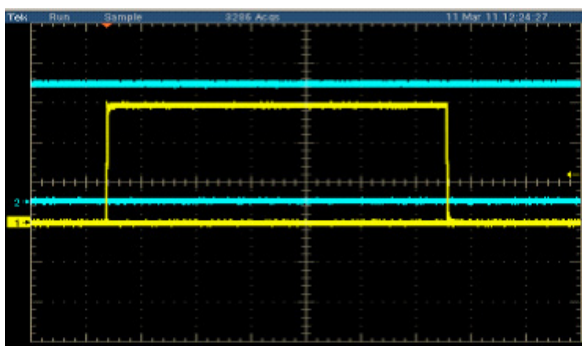


Figure 5: When the blue signal needs to realize the interlock function, it becomes high level, and the yellow signal is shutdown it just becomes low level.

DATA TRANSMISSION FUNCTIONS

The timing system prototype also considered the data transfer function, such as time-stamping, machine run mode data adn so on. EPICS-based control system can be used NTP (Network Time Protocol) to synchronize the time of each IOC. If the time granularity is the repetition period (such as J-PARC is 50Hz, U.S. SNS is 60Hz), then the timing system can be used to give each IOC to provide such high-precision time synchronization[3]. If the accelerator beam need to define some of the machine transfer mode and beam mode, in a different mode, some devices have different time charged action. We therefore need a model before the start of the model number will be sent to the running time of each site. Each time the site received mode information, to register the corresponding with different parameters, in order to achieve this mode of timing.

Data transfer operation mode is the accelerator data and time stamp and other information through the timing system to send and receive, this study we use the Swiss light source driver to Achieved. The driver contains four EPICS application: mrfApp, mrfDataBufferApp, regDevApp and generalTimeApp[4].

The four EPICS application to achieve three main functions: (1) mrfApp timing system to achieve the basic functions, which provides the trigger signal and clock signal; (2) mrfDataBufferApp and regDevApp accelerator operating mode to achieve data transmission;(3) generalTimeApp achieve timestamp (Timestamp) transmission.

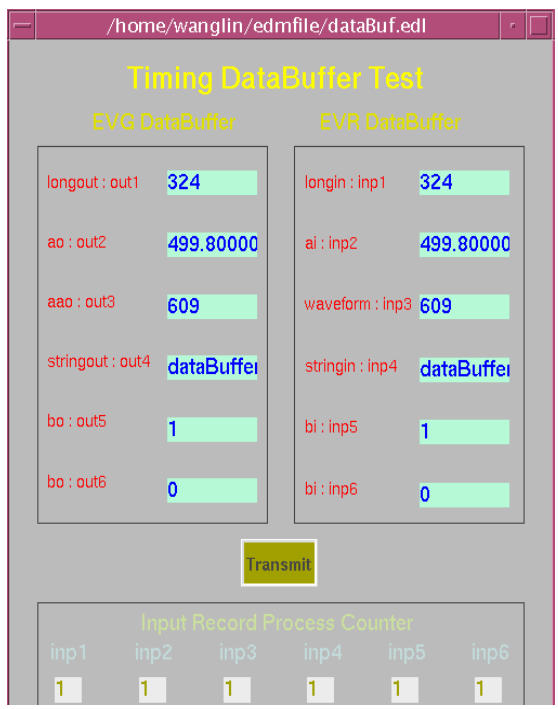


Figure 6: Data transfer test.

OPI

To facilitate the operation, in particular, produced a man-machine interface.The OPI has 4 part as shown in Figure 6 [5]. The first part is distribute bus&pulse output. The second part is CML output which showed in Figure 3. The 3rd is interlock function which showed in Figure 4. Last part is data transmission function. For the different mode, the right area will show different place which is correspond the left area.

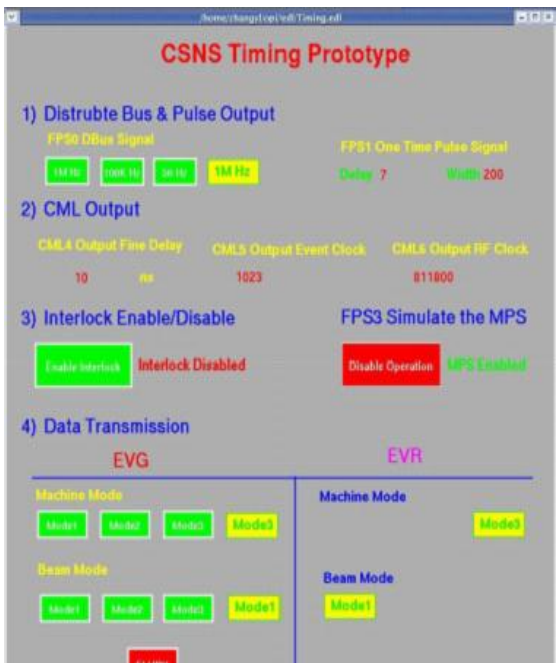


Figure 7: OPI of the prototype.

FOLLOW-UP

In CSNS,timing system is designed to provide triggers and clocks to the following systems,such as Front end, linac RF, injection, beam instrumentation, magnet power supply, RCS RF, extraction, spectrameter and target.Detailed discussions with the above systems have been done, while there are still some points need to be studied further.

REFERENCES

[1] Ge Lei, “CSNS timing system prototype Research report concludes”, Apr. 2011
[2] Guanglei Xu, Ge Lei, “New feature of timing system”, Aug. 2009.
[3] Peng Zhu, Ge Lei, Guanglei Xu,“CSNS timing system prototype programming and testing ”,Apr. 2011
[4] Lin Wang,“CSNS timing system prototype- data transmission”, Apr. 2011
[5] Yu Liang Zhang,“CSNS timing system prototype - interlock function and fine delay”, Apr.2011

THE BPM DAQ SYSTEM UPGRADE FOR SUPERKEKB INJECTOR LINAC

M. Satoh#, F. Miyahara, T. Suwada, and K. Furukawa, Accelerator Laboratory, High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan

T. Kudou and S. Kusano, Mitsubishi Electric System & Service Co., Ltd, 2-8-8 Umezono, Tsukuba, Ibaraki 305-0045, Japan

Abstract

The KEK electron/positron linac is a 600-m-long injector that provides the beams with different energy to four independent storage rings. The non-destructive beam position monitor (BPM) is an indispensable diagnostic tool for a long-term stable beam operation. In the KEK linac, about one hundred BPMs with the four strip-line type electrodes are utilized for the beam orbit and charge measurement. The measured beam orbit data is used for the beam orbit and energy feedback loops. The current data acquisition (DAQ) system for BPM comprises 24 fast digital oscilloscopes. They can work as a WindowsXP-based EPICS IOC.

Toward the SuperKEKB project, the upgrade of injector linac is going on for increasing the beam intensity and reducing the emittance. For the SuperKEKB injector linac, the electron beam emittance will be reduced one-fifth smaller than that of former KEKB project by using a new RF gun. For this reason, the measurement precision of BPM is strongly required to be increased. In this paper, we present the upgrade plan and status of DAQ system for BPM towards SuperKEKB project in detail.

INTRODUCTION

The KEK linac sequentially provides the electron and positron beams with different energies and intensities for four independent storage rings as shown in Table 1. For increasing the integrated luminosity and stored current stability, the simultaneous injection between KEKB electron and positron rings has been strongly required. In addition, the PF top-up injection has been also strongly demanded even during the KEKB injection. For these reasons, the injector upgrade project started in 2004 so that the simultaneous top-up of KEKB electron/positron and PF rings. This upgrade was completed in April 2009, and the simultaneous top-up injection among three independent rings was successfully achieved [1, 2, 3, 4].

Whereas the KEKB project has completed in the summer of 2010, the Super KEKB project has started for aiming at the peak luminosity of 40 times higher than that of former KEKB project. For this purpose, the injector linac upgrade is going on for increasing the beam intensity and reducing the emittance. In this linac upgrade, main issues are the construction of positron damping ring, the development of the new positron capture system for increasing the positron charge of four times present, and the installation of a low emittance electron gun as shown in Fig. 1. The performance required of beam position measurement is a higher precision of ten micro meters or less since the emittance of electron and positron beams are less than 20 mm·mrad.

masanori.satoh@kek.jp

Table 1: Injection Beam Energy and Charge for Each Ring

	KEKB e-/SuperKEKB e-	KEKB e+/SuperKEKB e+	PF	PF-AR
Injection beam energy (GeV)	8/7	3.5/4	2.5	3
Beam charge /bunch (nC)	1/5	1 (10*)/4(10*)	0.1	0.1

*Charge of primary electron for positron production

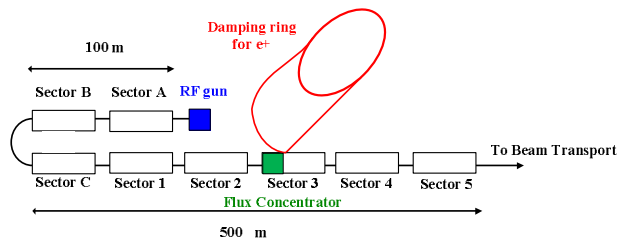


Figure 1: Schematics drawing of SuperKEKB injector linac. The coloured parts will be newly installed for SuperKEKB project.

LOW EMITTANCE ELECTRON BEAM DELIVERY

For the SuperKEKB project, the emittance of positron beam will be reduced 10 mm·mrad from 2100 mm·mrad by using the damping ring newly constructed. The

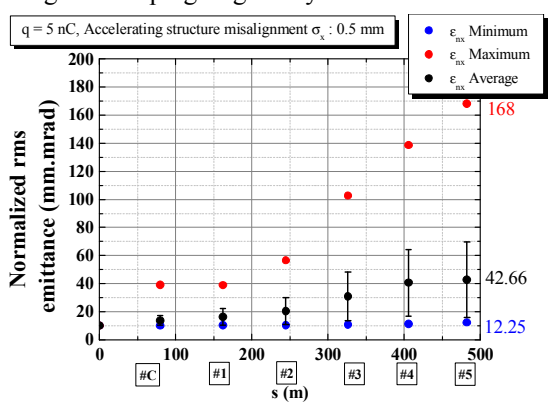


Figure 2: Simulation result of horizontal emittance growth in the KEK injector linac from Sector C to Sector 5.

emittance of electron beam will be reduced 20 mm·mrad from 100 by a low emittance rf gun. The low emittance electron beam should be delivered to the ring without damping ring due to cost reduction. In the high intensity electron linacs, emittance growth can be caused by the misalignment of accelerating structures and magnets.

Figure 2 shows the simulation result of horizontal emittance growth in the KEK linac. Here, it is assumed that the KEK injector linac configuration from Sector C to Sector 5, bunch charge of 5 nC and initial normalized rms emittance of 10 mm·mrad, and the accelerating structure misalignment of 0.5 mm with standard deviation. In this simulation, the emittance calculations were carried out by using simulation code elegant and 100 different error seeds. The simulation result shows that the average and maximum emittances at the end of Sector 5 are 42.66 mm·mrad and 168 mm·mrad, respectively. The final emittance strongly depends on the seeds of error.

On the other hand, the fine control of beam orbit can cure the emittance deterioration. Figure 3 shows the simulation result of emittance growth compensation by using the fine control of initial beam offset and angle. In this simulation, we used the seed of alignment error corresponding to the maximum emittance growth of 168 mm·mrad. The simulation result shows that the final emittance can be reduced 11.5 mm·mrad from 168 mm·mrad by adjusting of initial beam angle and offset. The control values of initial beam angle and offset are 100 μrad and 10 μm, respectively. From these simulation results, the high precision beam position measurement is strongly required for the fine beam orbit control of SuperKEKB injector linac.

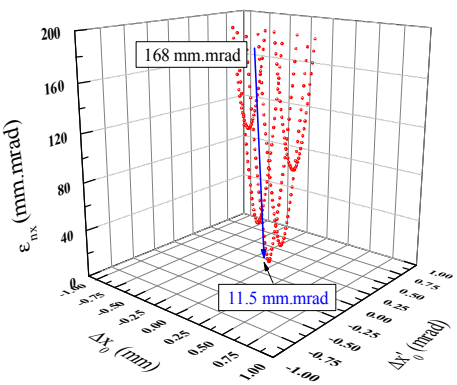


Figure 3: Simulation result of emittance growth compensation by using beam orbit control.

BPM DAQ SYSTEM

Present System

In the KEK linac, many kinds of feedback loops have been developed and utilized to stabilize the beam orbit, energy, and energy spread [5, 6, 7]. These feedback loops make use of the beam position information acquired by the non-destructive BPMs [8]. About one hundred stripline-type BPMs have been installed in the KEK linac.

The twenty four front-end systems have been installed in the linac klystron gallery at a nearly equal interval along the beam line. The each DAQ system deals with the analogue signals of 3 to 6 BPMs. A schematic drawing the present DAQ system is shown in Fig. 4. It comprises a fast digital oscilloscope (Tektronix DPO7104; 10 GSa/s, 4 channels, 8 bits, CPU P4/3.4 GHz, Gigabit-Ethernet) and a cable combiner box.

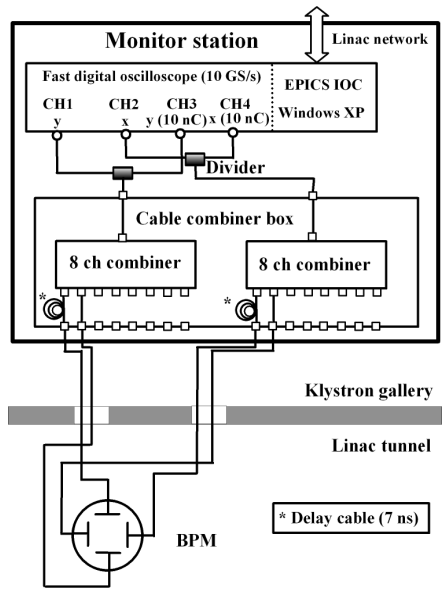


Figure 4: Schematic drawing of the present BPM DAQ system.

The four signals coming from one BPM are fed to two signal combiners (vertical and horizontal) together with the signals from other BPMs. The delay cables corresponding to a 7 ns time delay are used to avoid waveform overlaps at the signal combiners. The each output of combiner box is divided again into two signals. Since it is impossible to change the vertical scale of oscilloscope in every 50 Hz, CH1/CH2 and CH3/CH4 are used for the low charge and the high charge modes, respectively. The waveforms digitized at a sampling rate of 10 GSa/s are analyzed and converted into the beam parameters (beam charge, horizontal position, vertical position), taking into account the calibration coefficients. The DAQ software has been developed by using Microsoft Visual Studio 2005 C++, TekVisa, and EPICS R3.14.8.2 libraries. The DAQ software is running on the fast oscilloscope, and each DAQ can work as an EPICS IOC [9]. The similar DAQ system is also utilized for the KEKB and PF-AR beam transport lines [10].

The beam position measurement precision of the present system is listed in Table 2. These figures show the experimental results measured by three BPMs method [5]. The result of 1 nC shows worse precision than that of 0.1

nC since the same vertical scale settings are used for the measurements of 1 nC and 7 nC. There, the maximum ADC counts of 1 nC are smaller than the case of test with 0.1 nC electron. From these experimental results, it is difficult for the present DAQ system to achieve the beam position measurement precision of 10 μm or better. Toward Super KEKB linac, a new BPM DAQ system should be implemented since the ADC resolution of 8 bits limits the position measurement precision in the present system.

Table 2: BPM Precision of the Present DAQ System

Beam charge (nC)	Measurement precision (mm)
0.1	0.06626
1	0.10645
7	0.05052

New System

For the new DAQ system, the measurement precision of about 10 μm or better is one of key requirement. In addition, all beam position should be measured in every 20 ms interval. The SuperKEKB injector linac should perform the simultaneous injection to four independent rings since the beam lifetime of SuperKEKB is much shorter than that of KEKB rings. In the current design, it is estimated to be around 10 minutes. Figure 5 shows the beam operation scheme of SuperKEKB injector linac. The electron/positron beams with different energies and amounts of charges are delivered in every 20 ms. For this reason, the fast and precise attenuation control is also key requirement for the new system.

We adopt Libera Brilliance Single Pass unit as a candidate of the new system [11]. It is a commercial product of Instrumentation Technologies. It is a module dedicated for BPM DAQ and widely used at many recent accelerator facilities. Libera Brilliance Single Pass unit comprises four 16-bits ADCs of 125 MHz sampling frequency, two pass band saw filters in each analogue channel, a single-board computer (SBC), a field-programmable gate array (FPGA), and so on. It has also the variable attenuator with maximum attenuation of 31 dB. The attenuation level can be adjusted in 1 dB step.

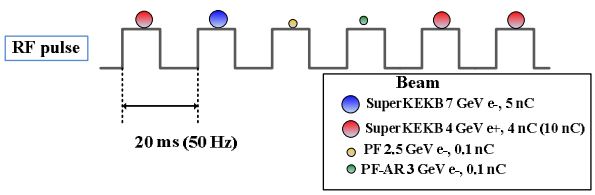


Figure 5: Schematics drawing of SuperKEKB injector linac beam operation. The beams with four different energies and amounts of charges should be managed.

The typical analogue signal shape from BPM electrode is a bipolar of 3 ns in width. Applying double saw filters, the signal is stretched to around 100 ns as shown in Fig. 6. After about 10 data points are sampled, the sum of squares of them is used as signal amplitude from each electrode. The data processing of digitized waveform is carried out on the FPGA side. The calculated beam position and other data are transferred to SBC side which is the embedded Linux operation system with fast Gigabit Ethernet. A client software like beam orbit display can retrieve the information of beam position and amount of charge via EPICS CA protocol.

EVALUATION OF DAQ PERFORMANCE

We evaluate the basic performance of Libera Brilliance Single Pass unit. Figure 7 shows the result of data acquisition defect by using test pulse of 50 Hz while four hours of continuous measurement. In this graph, the timestamp intervals between previous and current acquisitions are plotted as a function of test pulse number. From this result, all beam pulse can be measured up to 50 Hz since all timestamp differences are less than 40 ms. The more long-term experiment will be also carried out soon.

Figure 8 shows the precision of beam position measurement by using 1000 successive pulses of 0.1 nC electron beam. In this experiment, we used the eleven different steering settings and three LIBERAs connected to different BPMs. These results include the beam position fluctuation. The result of vertical direction at steering setting #5 of BPM#1 shows the standard deviation of 15 mm. The measurement precision of

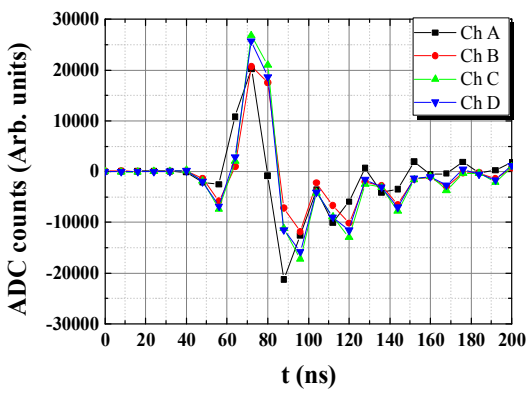


Figure 6: Raw waveform data measured by LIBERA Brilliance Single Pass unit.

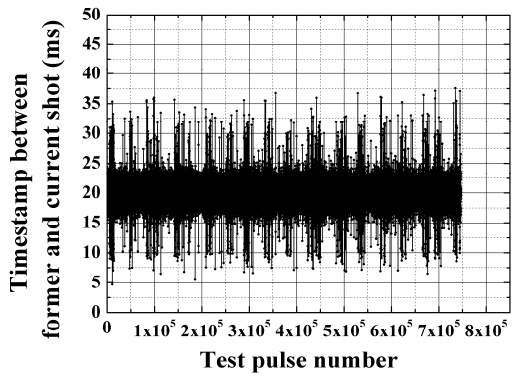


Figure 7: Result of data acquisition defect by using test pulse of 50 Hz while four hours of continuous test.

LIBERA Single Pass unit could be less than 15 mm when the effect of beam fluctuation is subtracted from this result. From these results, it is convinced that LIBERA Single Pass unit is feasible for our purpose.

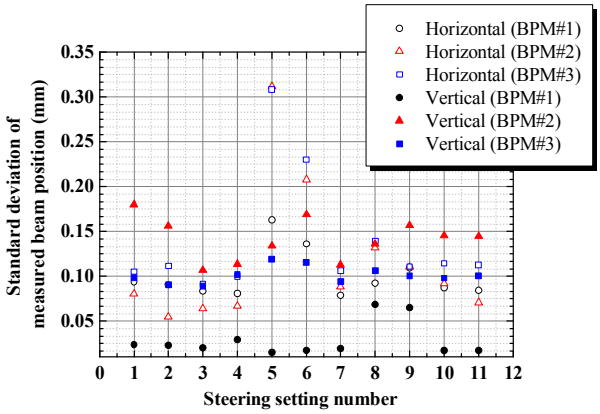


Figure 8: Precision of position measurement by using 0.1 nC beam and eleven different steering settings.

SUMMARY AND FUTURE PLAN

Toward the SuperKEKB project, the injector linac upgrade is going on for increasing the beam intensity and reducing the beam emittance. For the low emittance electron beam transport without damping ring, high precision beam position measurement and control are strongly required. For this purpose, the current BPM DAQ system should be replaced by new one with higher measurement precision. Recently, we preliminary evaluate the performance of a candidate Libera Brilliance Single Pass unit of Instrumentation Technologies.

In this performance test, we measured the speed performance of BPM signal detection and the precision of beam position measurement by using the KEK linac beam of 0.1 nC. The test result shows the speed performance is enough high for our application of 50 Hz beam measurement. The possibility of position measurement precision less than about 15 μm is also confirmed by the beam test.

The functionality of fast and precise attenuation control will be implemented soon. The complete test of position measurement precision by three BPMs method will also be carried out by using the KEK linac beam in the near future. At the same time, other schemes for getting the higher precision are also under consideration.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Rok Hrovatin, Mr. Matjaz Znidarcic, Mr. Peter Leban of Instrumentation Technologies for their fruitful discussions.

REFERENCES

[1] M. Satoh et al., "FIRST SIMULTANEOUS TOP-UP OPERATION OF THREE DIFFERENT RINGS IN KEK INJECTOR LINAC", in Proceedings of Linear Accelerator Conference LINAC2010, Tsukuba, Japan, pp.703-707 (2010).

[2] K. Furukawa, M. Satoh, T. Suwada, T. T. Nakamura, T. Kudou, S. Kusano, T. Nakamura, A. Kazakov, "NEW EVENT-BASED CONTROL SYSTEM FOR SIMULTANEOUS TOP-UP OPERATION AT KEKB AND PF", in Proceedings of ICALEPCS2009, Kobe, Japan, pp.765-767 (2009).

[3] Y. Ohnishi, T. Kamitani, N. Iida, M. Kikuchi, K. Furukawa, M. Satoh, K. Yokoyama, and Y. Ogawa, " Design and Performance of Optics for Multi-energy Injector Linac", in Proceedings of XXIV Linear Accelerator Conference, Victoria, British Columbia, Canada, Sep. 29–Oct. 3, pp.413-415 (2008).

[4] N. Iida, K. Furukawa, M. Ikeda, T. Kamitani, M. Kikuchi, E. Kikutani, M. Kobayashi, T. Mimashi, T. Mitsuhashi, T. Miura, Y. Ogawa, Y. Ohnishi, S. Ohsawa, M. Satoh, M. Suetake, T. Suwada, M. Tawada, A. Ueda, Y. Yano, K. Yokoyama, and M. Yoshida, "PULSE-TO-PULSE SWITCHING INJECTION TO THREE RINGS OF DIFFERENT ENERGIES FROM A SINGLE ELECTRON LINAC AT KEK", in Proceedings of the Particle Accelerator Conference, Vancouver, Canada, May 4–8, pp.2769-2771 (2009).

[5] T. Suwada, M. Satoh, and K. Furukawa, "Nondestructive beam energy-spread monitor using multi-strip-line electrodes", Phys. Rev. ST Accel. Beams 6, 032801 (2003).

[6] T. Suwada, M. Satoh, and K. Furukawa, "New energy-spread-feedback control system using nondestructive energy-spread monitors", Phys. Rev. ST Accel. Beams 8, 112802 (2005).

[7] K. Furukawa et al., "Beam Feedback Systems and BPM Read-Out System for the Two-Bunch Acceleration at the KEKB Linac", in Proceedings of the ICALEPCS2001, San Jose, November 2001, pp.266-268 (2001).

[8] T. Suwada, N. Kamikubota, H. Fukuma, N. Akasaka, and H. Kobayashi, "Stripline-type beam-position-monitor system for single-bunch electron/positron beams", Nuclear Instruments and Methods in Physics Research **A** 440, pp.307-319 (2000).

[9] M. Satoh, T. Suwada, K. Furukawa, Y. Hu, T. Kudou, S. Kusano, "EPICS IOC OF WINDOWSXP-BASED OSCILLOSCOPE FOR FAST BPM DATA ACQUISITION SYSTEM", in Proceedings of ICALEPCS2009, Kobe, Japan, pp.567-569 (2009).

[10] T. Aoyama, T. Nakamura, K. Yoshii, N. Iida, M. Satoh, and K. Furukawa, "UPGRADE OF READOUT SYSTEM FOR BEAM POSITION MONITORS IN THE KEKB BEAM TRANSPORT LINE", in Proceedings of ICALEPCS2009, Kobe, Japan, pp.495-497 (2009).

[11] http://www.i-tech.si/

IMPROVEMENT OF THE ORACLE SETUP AND DATABASE DESIGN AT THE HEIDELBERG ION THERAPY CENTER

K. Höppner*, Th. Haberer, J. M. Mosthaf, A. Peters,
HIT, Heidelberg Ion Therapy Center, University Hospital, Heidelberg, Germany
M. Thomas, A. Welde, Eckelmann AG, Wiesbaden, Germany
G. Fröhlich, S. Jülicher, V. RW Schaa, W. Schiebel, S. Steinmetz,
GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

Abstract

The HIT (Heidelberg Ion Therapy) center is an accelerator facility for cancer therapy using both carbon ions and protons, located at the university hospital in Heidelberg. It provides three therapy treatment rooms: two with fixed beam exit (both in clinical use), and a unique gantry with a rotating beam head, currently under commissioning. The backbone of the proprietary accelerator control system consists of an Oracle database running on a Windows server, storing and delivering data of beam cycles, error logging, measured values, and the device parameters and beam settings for about 100,000 combinations of energy, beam size and particle rate used in treatment plans. Since going operational, we found some performance problems with the current database setup. Thus, we started an analysis in cooperation with the industrial supplier of the control system (Eckelmann AG) and the GSI Helmholtzzentrum für Schwerionenforschung. It focused on the following topics: hardware resources of the DB server, configuration of the Oracle instance, and a review of the database design that underwent several changes since its original design. The analysis revealed issues on all fields. The outdated server will be replaced by a state-of-the-art machine soon. We will present improvements of the Oracle configuration, the optimization of SQL statements, and the performance tuning of database design by adding new indexes which proved directly visible in accelerator operation, while data integrity was improved by additional foreign key constraints.

THE HIT MEDICAL ACCELERATOR

The heavy ion accelerator at HIT is used for raster-scanning radiation of cancer patients (cf. [1, 2] for an overview) with different types of ions from two sources (upgrade to three sources in progress [3,4]) in several treatment rooms, two with horizontal fixed beam exit and the heavy ion gantry with rotatable beam exit (under comissioning [5]), and a beam exit for experiments (see Fig. 1). Each combination of source and destination may be used for medical treatment, represented within the Accelerator Control System (ACS) by the so-callled *virtual accelerator* number. A radiation plan consists of a series of beam pulses chosen from a catalogue of 255 different energy values (88–430 MeV/u for carbon, 48–220 MeV/u for protons), 6 focus sizes, 15 intensity values (2×10^6–5×10^8 particles per second for carbon, 8×10^7–2×10^{10} pps for protons), and 36 exit angles in case of the gantry. These tuples of beam settings are named the MEFI combinations. Both the virtual accelerator as the MEFI combination may be changed from beam pulse to beam pulse (multi-plexed operation).

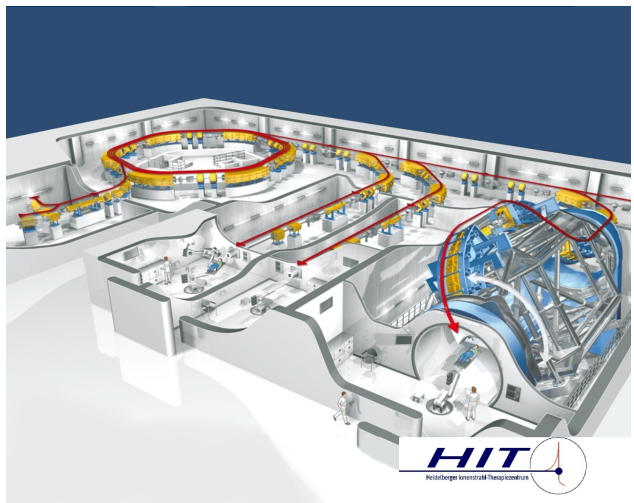


Figure 1: HIT accelerator facility with two ion sources, linear accelerator, synchroton, two horizontal beam exits and gantry for medical treatment. The experimental area is not shown.

DATABASE AS PART OF THE ACCELERATOR CONTROL SYSTEM

The database is a core part of the accelerator control system. It is used for

- a list of executed beam cycles for about 6 weeks together with measured values for e. g. currents of power supplies, data beam instrumentation (like ionization chambers), vacuum pressure, ...
- list of devices that are active for a virtual accelerator,
- settings for all accelerator devices to be written into the device controllers,
- alarm and log messages,

* klaus.hoeppner@med.uni-heidelberg.de

- users and access rights,
- computers and applications.

The device settings are generated as binary large objects (BLOBs) by the DSM, the data supply module, from the physical beam properties optimized for a representative number of MEFI values and interpolation over the full MEFI range. Finally, the device blob per MEFI is stored within the Oracle database.

There exist three kinds of device blobs to combine the need for adjusting the accelerator settings and test of different beam optics with the quality requirements for stable settings needed for medical treatment:

- flash blobs that are stored in non-volatile memory of the device controllers after verification for therapy,
- online blobs stored in the RAM of the device controllers that are changed during accelerator adjustment (without affecting the verified flash data), and
- offline blobs to save old device data, present in the database, only.

The checksums of the blobs stored in the device controllers are checked against the blob checksums stored in database on any beam request to ensure the integrity of the settings.

DATABASE PROBLEMS

Since about 2010 the existing Oracle database (Oracle 9i running on a server with a dual core Xeon CPU and 2 GB of RAM under Windows 2003) showed performance problems. This was both visible during therapy (when checking the blob checksums, writing measured values) as during accelerator adjustment (when writing generated online blobs into database, copying online blobs into flash blobs after verification).

Obviously, a major reason was the outdated hardware, that was upgraded to a modern blade center during summer 2011 (as presented in [6]), but also issues beyond hardware should be addressed. Thus, we established a working group consisting of HIT controls group, Eckelmann AG (as supplier of the proprietary control system) and the database experts from the accelerator controls department of the GSI Helmholzzentrum für Schwerionenforschung in Darmstadt, Germany.

The workgroup analyzed both the setup of the current Oracle installation (e. g. by statspack[1] reports of usual operating tasks) as well as the structural design of the database (tables, indexes, constraints).

RESULTS, RECOMMENDATIONS, AND REALIZATION

Hardware and Oracle Setup

As expected, both the CPU speed and the memory of the database server were considered insufficient. It was recommended to upgrade the Oracle installation to version 11g and use Oracle's Automatic Storage Manager (ASM) as file system which is optimized for the data storage of an Oracle instance. RMAN[2] should be used as backup tool instead of the previous daily tablespace exports.

With the installation of the new blade servers during the summer shutdown period, these proposals were put into practise.

Number of Commits

The statspack analysis showed a high number of about 20 commits per second. This is caused by several data gateway processes that insert measured values from the devices into the database. In the current realization of the control system, a commit command is sent per insert statement. It was proposed to decrease the number of commits per second.

Eckelmann AG was checking the impact of grouping several insert statements per commit on the overall performance of the ACS. Obviously, the longer the delay between insert and commit, the slower the GUIs will present state changes and measured values to the operator. Thus, a compromise had to be found. Additionally, since several applications are writing into the database, the effort to implement and test a better commit mechanism was rather high.

Finally, the changes will be installed at HIT in October.

Structure of Tables

Many requirements for accelerator operation changed until or even since the accelerator facility became operational for medical treatment, inspired by the lessons learnt during commissioning. Thus, there was an evolution of the ACS from its original design until its current state. The database design reflects this evolution.

It was the overall impression that the database design for rather static data (like definition of ion types, beamlines, list of devices and their assignment to beamlines) was good, with appropriate relations between tables defined by foreign key constraints. But in some cases, data columns with dynamic, volatile data were added into these "static" tables due to changing or adding new features for accelerator controls. Tables for dynamic data (beam cycle data, measured values, messages) were often lacking foreign key constraints.

The recommendations for an improvement of the database design were as follows:

- Separate tablespaces for static (configuration) data and dynamic (measured) data,
- avoid to mix columns with static and volatile data in tables,
- define proper relations between tables by adding foreign key constraints where missing,

[1] statspack is a performance monitoring and reporting tool provided by Oracle starting from version 8i, cf. `http://www.oracle.com` or `http://www.orafaq.com/wiki/Statspack`.

[2] RMAN is a tool provided by Oracle to backup, restore and recover Oracle databases, cf. `http://www.oracle.com`

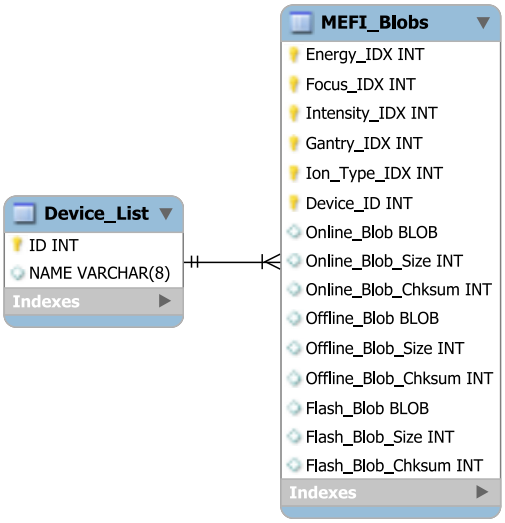


Figure 2: Current design of the table with MEFI device blobs, with three blobs per record.

- add indexes to optimize database queries,
- redesign "wide" tables to achieve a better in-memory-caching of data records within the Oracle database instance.

A good example for the last item is the table of device blobs per MEFI combination. As mentioned above, there exist online, offline and flash blobs. In the current table design, the table for MEFI device blobs has three blob columns (as shown in Fig. 2), one per blob type, i. e. when loading the data record for a MEFI combination and a device, three blobs are read and have to be cached in memory. Consider the workflow during accelerator adjustment, when online blobs are generated and written into database for all MEFI combinations and all devices. For downloading the new online blobs into the RAM of the device controllers, many records from the table with MEFI device blobs have to be read, and though only the online blobs are needed, the offline and flash blobs are read as well, consuming I/O ressources and memory of the database cache.

It was recommended to keep the blobs itself in an own table and replace the blob columns in the MEFI blob table by references. This would increase the number of tables itself, but keep the MEFI blob table compact. By appropriate joins between the MEFI blob table and the table(s) keeping the "real" blobs, the set of blobs that are read by Oracle would be reduced to those that are really needed. Figure 3 shows a possible redesign, where the blob columns are replaced by references to another table. Since changes to the table structure have impacts on the whole ACS, a redesign needs to be thorougly planned and tested. This process will be done until the end of this year.

Other recommendations listed above are already implemented or will be installed during the next ACS update in October 2011 (like adding foreign key constraints). The separation of static and volatile data in different tablespaces was realized during the migration from Oracle 9i to 11g.

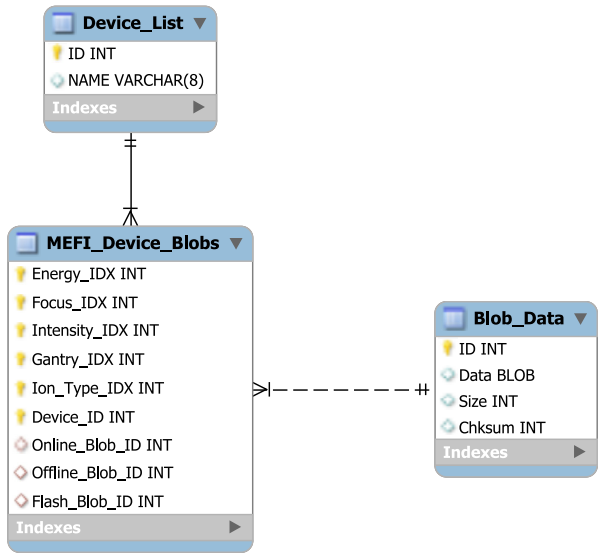


Figure 3: Possible change of table structure for MEFI device blobs by replacing the blob columns with a reference to another table keeping the blobs.

Several missing indexes were pointed out by the working group, that were added in the meantime, and some indexes were reorganized to achieve unique indexes as unique index scans are more efficient than normal index scans.

FIRST RESULTS

In combination with the new hardware platform, the changes to the Oracle database show already a positive performance effect.

Figure 4 compares the time between the stop and start broadcast signals (as sent by the central timing system), i. e. the dead time between beam cycles needed to prepare and setup the next cycle, for a week of therapy in September 2010 and September 2011, respectively. While the time

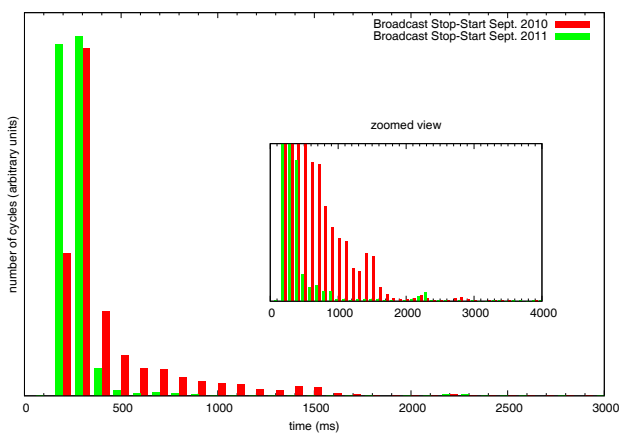


Figure 4: Comparison of dead time between beam cycles needed for preparation of the next beam pulse: September 2010 (red) and September 2011 (green).

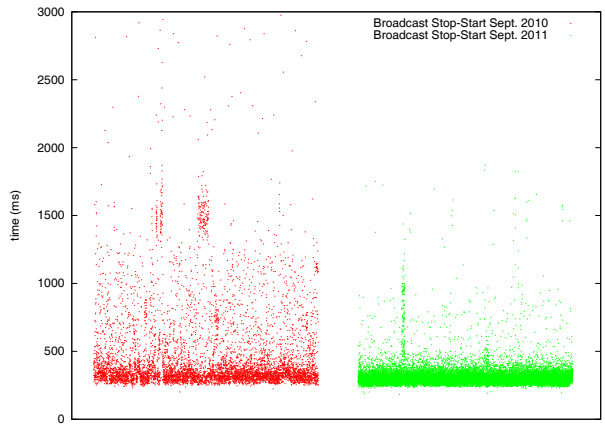


Figure 5: Scatter plot of dead time between beam cycles needed for preparation of the next beam pulse: September 2010 (red) and September 2011 (green).

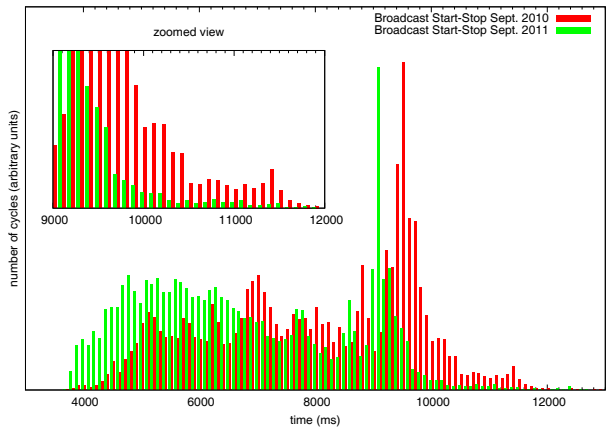


Figure 6: Comparison of beam cycle time (for up to 5 s tumor radiation): September 2010 (red) and September 2011 (green).

needed to prepare the next beam cycle after the end of the previous cycle is about 250 ms for most of the cycles, the average dead time was spoiled by many cycles with a much longer preparation time before the start broadcast signal was sent. The analysis of traces of servers and device controllers showed, that this effect was mainly caused by

- a delay in the Oracle database when creating the database record for the upcoming cycle and querying the list of devices that are active for the cycle,
- a read delay in the flash memory of the device controllers when checking the MEFI blob checksums.

After both problems were solved this year, it is immediately visible from the plot that most delayed cycles were eliminated, with the effect that the average preparation time was decreased from 490 ms in September 2010 to 390 ms in September 2011. The scatter plot in Fig. 5 also shows, that the number of cycles with preparation time above about 400 ms is decreased dramatically.

We compared also the times between the broadcast signals for cycle start and stop during therapy (based on medical treatment with up to 5 s radiation of the tumor per beam cycle) for September 2010 and 2011. As seen in Fig. 6 the cycle performance increased a lot, caused both by improvements in the therapy control system (TCS) and the communication between ACS and TCS, as well as the performance tuning of the database. The analysis of the traces showed that the time needed for updating the cycle status in the database was sometimes up to 200 ms.

CONCLUSION

As presented in the previous section, the accelerator performance was increased a lot during the last year. This is due to many efforts to optimize the ACS, both by hardware as well as by improvements to the database.

The improvements are also visible in GUIs for the operator that profit from better database querying performance, and time consuming tasks during accelerator adjustments, like interpolating and generating MEFI blobs or flashing are much faster than before. E. g., an interpolation of all devices for the gantry needs about 1.5 hours per ion type, instead of 6 hrs. before.

The recommendations didn't just address database performance, but data integrity, too. The realization of proper relations between the tables, as it is currently in progress or preparation, will help to avoid both redundant or even invalid data within the database.

REFERENCES

[1] Th. Haberer et al., "The Heidelberg Ion Therapy Center", Rad. & Onc., 73 (Suppl. 2), 186–190, (2004).

[2] S. Combs et al., "Particle therapy at the Heidelberg Ion Therapy Center (HIT)", Radiotherapy and Oncology, Vol. 95, 41–44, (2010).

[3] A. Peters et al., "Operational Status and Further Enhancements of the HIT Accelerator Facility", IPAC 2010, Kyoto, Japan, May 2010, MOPEA006, http://www.jacow.org.

[4] R. Cee et al., "Status of the Ion Source and RFQ Test Bench at the Heidelberg Ion Beam Therapy Centre", IPAC 2011, San Sebastián, Spain, September 2011, WEPS044, http://www.jacow.org.

[5] M. Galonska et al., "Commissioning of the Ion Beam Gantry at HIT", IPAC 2011, San Sebastián, Spain, September 2011, THOAB03, http://www.jacow.org.

[6] J. M. Mosthaf et al., "Upgrade of the Server Architecture for the Accelerator Control System at the Heidelberg Ion Therapy Center", ICALEPCS 2011, Grenoble, France, October 2011, MOMMU009, http://www.jacow.org.

DID WE GET WHAT WE AIMED FOR 10 YEARS AGO?

A. Augustinus, P. Chochula, L. S. Jirdén, M. Lechman, P. Rosinský, CERN, Geneva, Switzerland
O. Pinazza, INFN Sezione di Bologna, Bologna, Italy and CERN
G. De Cataldo, INFN Sezione di Bari, Bari, Italy and CERN
A. N. Kurepin, INR INR RAS – Institute for Nuclear Research of the Russian Academy of Sciences, Moscow, Russia
A. Moreno, Universidad Politécnica de Madrid, ETSI Industriales, Madrid, Spain

Abstract

The ALICE Detector Control System (DCS) is in charge of control and operation of one of the large high energy physics experiments at CERN in Geneva. The DCS design which started in 2000 was partly inspired by the control systems of the previous generation of HEP experiments at the LEP accelerator at CERN. However, the scale of the LHC experiments and the use of modern, "intelligent" hardware and the harsh operational environment led to an innovative system design. The overall architecture has been largely based on commercial products like PVSS SCADA system and OPC servers extended by frameworks. Windows has been chosen as the operating system platform for the core systems and Linux for the front-end devices. The concept of finite state machines has been deeply integrated into the system design and the design principles have been optimized and adapted to the expected operational needs. The ALICE DCS was designed, prototyped, and developed at a time when no experience with systems of similar scale and complexity existed. At the time of its implementation the detector hardware was not yet available and tests were performed only with partial detector installations. In this paper we analyse how well the original requirements and expectations set ten years ago comply with the real experiment needs after two years of operation. We provide an overview of system performance, reliability and scalability. Based on this experience we assess the need for future system enhancements to take place during the LHC technical stop in 2013.

INTRODUCTION

The design of the ALICE DCS started late 2000. In its initial phase the project was able to profit from already advanced developments in other LHC experiments and from tools and guidelines provided by the Joint Controls Project (JCOP) [1]. The existing concepts were compared with ALICE detector needs and from this the first ALICE DCS architecture was designed [2]. The first presentation given to the ALICE Technical Board in 2001 defined the roadmap for the whole DCS. In the following chapters we will review the presented key concepts and compare them with the current system implementation.

ALICE DCS SYSTEM CONTEXT

The architecture of the ALICE DCS is strictly hierarchical. The top level is formed by the central DCS which coordinates the individual detector systems. Each detector system is then divided into sub-systems which group devices with similar functionality. Subsystems are then further partitioned into devices, modules, and channels according to individual detector architectures. Each component of this hierarchy is modelled as a finite state machine (FSM) with standardized states and recognized commands. The commands are propagated through the hierarchy from parent to child, while the states are reported by children back to parents. The global status of the DCS takes always into account the states of all children in the hierarchy, with exception of any which have been masked (excluded) by the experts. This approach has been implemented using the SMI++ toolkit, which proved to be an extremely powerful, flexible and reliable component.

The core of the ALICE DCS is based on a commercial SCADA system – PVSS II [3] – which is extended by frameworks developed at CERN. Individual PVSS systems are grouped by detector and supervised by central DCS system.

The whole system is configured using data stored in the central ORACLE database. Up to 6GB of configuration data is uploaded to detectors before a physics run can be started.

Acquired data is compared with predefined operational limits and automatic or operator driven actions are taken in case of an anomaly. A subset of the acquired data is stored in the ORACLE database for further use in offline analysis or by detector experts.

A large amount of data is being exchanged between the central DCS and external systems such as electricity, magnet control, gas systems, cooling, etc. Data exchange between ALICE and the LHC is provided within the DCS context and currently represents one of the major data processing challenges.

Finally, the DCS communicates with all components of the ALICE online and offline system. The information exchange is mostly based on the FSM states and commands, but a large amount of data also flows via dedicated data publishers and file exchange servers.

COHERENT AND HOMOGENOUS SYSTEM

The DCS covers a large number and variety of devices, systems and components, which are developed by various groups in parallel. The role of the central team is the coordination and monitoring of these developments. Standards and recommendations are issued and reviewed regularly. Whenever possible, common solutions are recommended and deployed both in the hardware and software domains [4].

In the original concept, the DCS backend, consisting currently of about 170 computers, was based on the Windows platform, with a very limited number of Linux installations. With the growing complexity of the overall DCS systems, the number of Linux services increased to about 30% of all installed systems. Another increase in Linux installations is related to the front-end boards, which contain embedded operating systems. In the full configuration there will be ~800 of such boards installed in ALICE, making Linux the main OS platform.

In the ALICE DCS the operating system flavours and software versions are defined centrally. The developers are then requested to follow these requirements. Due to continuous system evolution, the number of required software packages is increasing and there are several justified deviations from the standards. The central team carefully monitors the exceptions and updates standards as needed. The software upgrades are usually carried out during the longer LHC breaks in the winter period.

FLEXIBLE AND SCALABLE SYSTEM

The control system has been operational since the first device installations. It follows detector evolution with the goal to cover the lifetime of the experiment. The system architecture has to be flexible to accommodate future detector developments and operational procedures. At the same time, the system needs to be scalable to cope with the experiment's growth.

Keeping these requirements in mind, the DCS has been designed to accommodate possible future extensions. The first architectural principle was to build a modular system and avoid a monolithic architecture. Separate and independent control systems have been built for each detector, avoiding cross-detector dependencies. The individual detector systems consist of one or several sub-systems which control devices with similar functionality. All detector systems are then integrated into the global controls system, using the distribution features of PVSS.

On the lowest level, the device access has been strictly separated from the control tasks, which are implemented in PVSS. Modules communicating with devices recognize simple commands, but do not implement any control logic. On the supervision level, the operator is presented with a set of tools which allow for experiment operation. The tools are executed on computers separated from the machines in charge of the device control. The clear advantage of the modular approach is the possibility to distribute the individual parts across several computers. This allows for better resource sharing and assures scalability. Separation of operator tasks from the controls functionality helps to prevent a possible critical system overload, triggered, for example, by an excessive request – like conditions data retrieval for a long time period which could overload the operator machine, but will leave the controls functionality intact.

OPERATIONAL MODES AND CONCURRENT OPERATION

One main difference between DCS and other online systems is that the DCS needs to remain operational during all phases of the experiment. Full DCS functionality is expected also during the periods without data taking – shutdown periods, upgrades, etc. The requirements during the different periods vary; the DCS is designed to cope with them, successfully providing its services 365 days a year.

When experiment conditions allow for it, the detectors are able to get some autonomy in the operation of their hardware. Using the SMI tools, whole detectors, or their parts, can be excluded from the central DCS and released to local operators. Such systems do not receive commands from the central operator, but the alerts are still propagated and followed in a standard way.

The operational experience revealed one potentially weak point of this approach: the devices under the local control do not report their status to the central operator and ignore commands. This might be dangerous during the critical phases of the experiment – for example during the magnet ramp or injection of particles to the LHC. In such periods, the detector settings, such as high voltage or frontend configuration, must be compatible with the intended operation. For these reasons a new technique has been developed and deployed in parallel to the standard SMI tools. A set of software probes controls all critical settings and read back values, independent of the FSM status. Thanks to this, the safety of excluded devices can be assessed and taken into account regardless of their ownership within the hierarchy and assure the correct execution of DCS procedures.

The original hierarchical approach and the FSM standardization across all detectors as defined 10 years ago proved to be valid and successful. However, with the knowledge gained during the operation with LHC, many extensions were implemented and the whole system became too complex to be efficiently exploited by the shift crew. A set of tools were then developed to group several functionalities and to allow the operator to execute complex tasks without the in-depth knowledge of the underlying FSM processes.

USER FRIENDLY AND INTUITIVE OPERATION

During normal operation, only a small shift crew of 2-6 people control the whole experiment from the central workspace. Since the operators are not necessarily detector experts, special attention is given to the presentation of the system.

A dedicated component, ALICE DCS UI, has been developed and deployed in all detector systems. It allows for standardized presentation of the systems to operators; it provides a uniform look and feel and is easy to use. Experience has shown that the operators have a tendency to cover the monitor screens with many windows. The philosophy adopted in ALICE has therefore been to aggregate the information and to provide a summary to the operator who then can decide to browse the hierarchy for more details.

All critical operations which require operator action are executed from dedicated interfaces. These contain all the buttons and indicators gathered from other components which are required for the current task. The operator can therefore fully focus on the action, without the need for browsing the system to get information.

Although lots of attention has been given to intuitive operation, human errors were responsible for the majority of the operational incidents, which led to data taking delays. After careful analysis of all events, the tools were redesigned and simplified, with most of the actions automated. In 2010 we started to deploy expert systems which monitor the critical operations and inform the operator about all anomalies, with clear troubleshooting instructions. These instructions are displayed directly on the interface related to the executed task along with all indicators and buttons required for the problem resolution. Unlike the alert system, which typically informs about exceeded thresholds for a monitored value, the expert system proactively follows the operation and is able to detect events such as an incorrect sequence of commands that did not yet trigger an anomaly.

AVAILABLE, SAFE AND RELIABLE SYSTEM

Whereas the safety of the personnel is the task of the CERN Safety System, ensuring the integrity of the detector equipment is largely the task of the DCS. The control system allows for hardwired or software actions in case of hazardous situations. The system must be reliable and available; where needed, the equipment is running on safe power.

The current operational performance proves that the ALICE DCS reached this goal. While during the start-up phase of the experiment the shift crew required daily assistance of the central team, in 2011 we reached a stable state where the shift crew does not need to consult the expert for several weeks in a row. Besides the stability of the deployed software and hardware, a large contribution to the reliability and smooth operation comes from comprehensive training provided to the shifters.

SYSTEM MAINTAINABILITY

The complexity of the ALICE experiments clearly exceeds the capabilities of a small central team. The expertise related to individual detector operation is maintained in the various institutes that developed the detectors. By using well defined interfaces and standards it is possible to integrate individual developments into the overall DCS, however long term support and maintainability becomes a major concern.

The overall culture of the academic environment expects exploration and deployment of latest tools and technologies. A small central team is not able to certify all the proposed components and is therefore forced to insist on conservative solutions, providing the required functionality.

In the lifetime of a large experiment such as ALICE, the migration of experts is an inevitable fact which has to be taken into account. It is not uncommon that the major developments are carried out by graduate or PhD. students who leave and continue their career in other fields. The developed systems are transferred to new colleagues. The transfer procedure usually does not explain all the background and context, and the developers tend to focus on the solution without taking into account the overall system architecture. The role of the central team is to monitor this evolution and ensure that the takeover will respect the agreed rules and principles.

The key to a maintainable system lays in the standardization. The ALICE DCS project team imposed rules and common solutions whenever possible. From the very beginning, the detector requirements were carefully reviewed and standard solutions were proposed. Thanks to this approach, despite the large differences in detector architectures, we were able to limit the variety of devices used; only three brands of power supplies are used in ALICE, for example. The uniformity of the hardware has clear benefits for the support provided by the central team.

Operational experience gained in past years has also proven the success of the deployment of common solutions also in other subsystems, such as gas or cooling. One exception to this model is however the front-end and readout electronics (FERO). By the time of the DCS design, the individual FERO architectures were already significantly advanced, with most of the modules already produced. The DCS had to cope with a large variety of architectures based on completely different solutions. A perfect example is the deployment of field buses. While in the power system we were able to restrict this layer to CANbus and Ethernet, the FERO access is based on JTAG, RS-232, VME, CANbus, Ethernet and a number of non-standard solutions developed in the institutes. To cope with this diversity the concept of Front-end Device (FED) was developed [5].

The FED architecture is inspired by the commercial OPC technology. The low level layer of the FED software is responsible for the communication with the FERO. Its

development requires deep expertise provided by detector designers, but once deployed it remains a relatively stable component, with changes linked mostly to hardware updates. The upper FED layer is based on the standard communication protocol DIM. Similar to OPC, it is implemented as a server that reacts to standardized commands sent by the clients and reports back the status. Several devices require a middle layer which translates the general commands into device specific actions. In most cases, the functionality of this layer focuses on sequencing and synchronizing commands transmitted directly to device channels, formatting and compression of data and low level error handling. The FED clients are implemented in PVSS. The FED concept provides a hardware abstraction layer, which allows for communication with the different FERO architectures in a unified way.

In the past few years, the FED concept has been used for other non-standard devices and subsystems in ALICE. Despite the obtained unified operation, FED remains one of the major challenges in ALICE DCS due to the complex functionality of its low-level layers.

OUTLOOK FOR SYSTEM EVOLUTION

During the long LHC technical stop planned for 2013, the DCS will profit from the gained experience and the system will be modernized.

As a first step, the standards are already being reviewed and new solutions will be proposed and made available to the detectors. All exceptions accumulated during the past years of operation will be removed and the system uniformity will be restored.

The plan is also to update all software components and to deploy the latest operating systems and tools. This is very delicate process as many commercial components do not follow the same evolution. Validation of the new systems has therefore already started in order to give sufficient time for finding satisfactory solutions.

The design of the detector hardware, including the computer interfaces, was launched more than 10 years ago. Stable operation of the experiment does not allow for regular upgrades. As a consequence, the DCS needs to operate a large fraction of obsolete hardware which in turn triggers a necessity to maintain a stock of spares. In some cases – like PCI interfaces, it is not possible to support the existing solution long-term. Therefore, new standards are being tested and prepared for deployment during the long technical stop.

CONCLUSIONS

The ALICE DCS project was launched 10 years ago. The construction, testing and operation phase of the system proved that the original assumptions and design principles were correct and efficient. The hierarchical approach and deployment of standards and common solutions contributed to the successful DCS operation.

The experience with the production system revealed many additional aspects that were not originally anticipated. One of the most visible examples is the front-end electronics with all its complexity and diversity. The DCS had to adopt already developed systems and build abstraction layers to allow for smooth integration and operation.

Interaction with systems external to DCS, increased data flow, and synchronization issues led to a need to create new unforeseen tools and procedures.

Comparing the original plans with the existing system it can be concluded that the system became more complex than anticipated, but the design principles have been well validated by operational experience. The main goal – to have a stable, reliable and safe system – has been reached. So, we got what we aimed for ten years ago.

REFERENCES

[1] Holme, O. et al., "The JCOP framework," proceedings ICALEPCS 2005, Geneva, Switzerland, 2005.

[2] ALICE Collaboration,Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger and Control System, CERN/LHCC/2003-062.

[3] ETM website http://www.etm.at/index_e.asp (the product was recently rebranded as "Simatic WinCC Open Architecture").

[4] Augustinus A. et al., "The wonderland of operating ALICE experiment", this Proceedings.

[5] Chochula P. et al., "Control and Monitoring of the Frontend Electronics in ALICE", Proceedings 9th Workshop on Electronics for LHC Experiments, LECC 2003, Amsterdam, The Netherlands.

RE-ENGINEERING OF THE SPRING-8 RADIATION MONITOR DATA ACQUISITION SYSTEM

T. Masuda#, M. Ishii, K. Kawata, T. Matsushita, C. Saji, JASRI/SPring-8, Hyogo, Japan

Abstract

We have re-engineered the data acquisition system for SPring-8 radiation monitors. The previous system consisted of dedicated digital indicator/controller modules (DICMs) optically linked with radiation monitors, embedded PCs for data acquisition, and programmable logic controllers (PLCs) for 1-h integrated dose surveillance. The embedded PCs periodically collected radiation data from the DICMs through GPIB. The DICMs and the dose surveillance PLCs were interfaced with an accelerator radiation safety interlock system (ARSIS). These components were dedicated, black-boxed, and complicated for operations. The GPIB interface was legacy and not reliable enough for the critical system. Therefore, we re-engineered the previous system by adopting PLCs and FL-net. Local PLCs were deployed as substitutes for all the previous embedded PC-based components. Another PLC was installed to enable centralized management of all 81 monitors. All the new PLCs and a VME computer for data acquisition were connected via FL-net networks. In this paper, the new system and a method for upgrading this system within the short shutdown interval of the accelerator operations are described.

INTRODUCTION

SPring-8 is the electron beam accelerator complex used in experiments that require a high-brilliance synchrotron radiation beam. According to the act on prevention of radiation disease, the radiation dose from the accelerator complex must be controlled. Therefore, it is mandatory for a radiation monitoring system to always monitor the radiation dose, and when the dose exceeds the alert level, it should immediately interrupt the accelerator operation through an accelerator radiation safety interlock system (ARSIS) [1]. A total of 81 radiation monitors including the 17 monitors used for the interlock system are deployed around the SPring-8 site.

The radiation monitoring system was very complicated because of the patched update one after another. It comprised two independent systems, namely a main system for 79 radiation monitors and a subsystem for two monitoring posts. The main system consisted of radiation monitors, optical transmitters with local controllers, optical converters to electrical serial communications, 79 digital indicator/controller modules (DICMs), 11 embedded PCs with GPIB and Ethernet interfaces, 3 programmable logic controllers (PLCs) to evaluate the 1-h integrated dose for the 17 interlock radiation monitors, and 3 PLCs for the interface with ARSIS. All the DICMs were mounted on eleven 19-inch racks scattered around the site. The DICMs monitored the radiation data on the basis of the three established alert/warning thresholds (low, high, and high-high). It also checked the normal operation of the radiation monitors and communication devices.

*masuda@spring8.or.jp

Although the radiation data were very important for the accelerator operation, these data could not be accessed in the same manner as other data related to the accelerator, because the radiation monitoring system was developed as the completely independent system with the accelerator control system. Therefore, we built the data acquisition system three years ago for the radiation monitors by installing a server PC that communicated with all the embedded PCs. We used the data acquisition software of message and database oriented control architecture (MADOCA) [2] to accumulate all the radiation data into the common database used for the accelerator control system. Figure 1 shows the previous data acquisition system used for the SPring-8 radiation monitors.

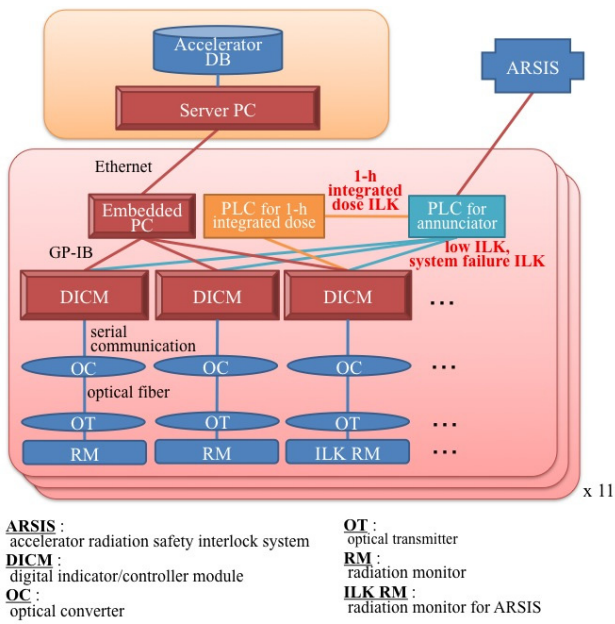


Figure 1: Previous data acquisition system for the SPring-8 radiation monitors.

PROBLEMS WITH PREVIOUS SYSTEM

Development of the previous data acquisition system enabled the accelerator operators to browse and investigate the status of the radiation monitors in the same manner as that used for the accelerators' data. The previous system, however, had the following problems.

- The GPIB interface employed in the previous system was not reliable enough for continuous use.
- The software that periodically collected data from the radiation monitors via the GPIB in the embedded

PCs was black-boxed. It was not practical to radically modify the software. However, it was necessary to improve the software because there was no way of knowing whether or not the software was running normally.

- Accelerator operators were unable to access the 1 h-integrated dose data used for radiation control, because these data were integrated by PLCs using the analogue outputs of the DICMS. These PLCs were patched at a later date and were independent from the data acquisition system.

Therefore, we radically re-engineered the data acquisition system to solve these problems.

NEW SYSTEM DESIGN

The new data acquisition system was designed in the following manner.

- We deployed local PLCs instead of the DICMs, embedded PCs, and GPIB interface to enhance the system reliability and to reduce the number of black boxes as much as possible.
- We integrated the independent subsystem for two monitoring posts into the new system.
- We assigned the integration function for dose data to the newly installed local PLCs.
- All the local PLCs were equipped with touch panels to enable user-friendly operations.
- We introduced a central PLC to centrally manage all the monitors near the central control room.
- We adopted FL-net [3], which was widely used in the SPring-8 and SPring-8 angstrom compact free electron laser (SACLA) control system, to connect all the local PLCs and the central PLC.

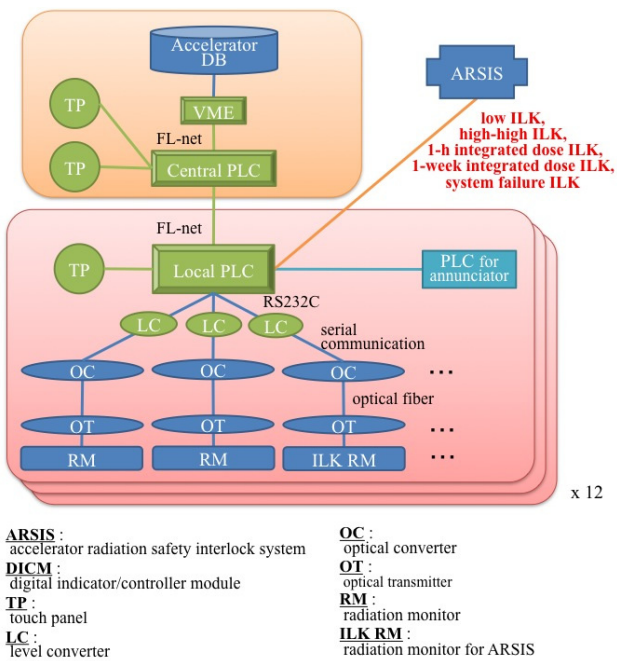


Figure 2: Schematic diagram of the new data acquisition system for the SPring-8 radiation monitors.

- The central PLC was provided with an additional FL-net module to enable connection with a VME computer that was linked to the accelerator control network. The VME computer collects all radiation data and incorporates them into the common database of the accelerator control system using the MADOCA framework.
- All the critical processes concerned with ARSIS are carried out by the local PLCs. The central PLC is not used for ARSIS.
- The interlock signals originating from the local PLC are sent directly to ARSIS. The existing PLCs are not used for interlock processing to simplify the system.

The new data acquisition system is shown in Figure 2. The new system consists of 12 local PLCs with touch panels, 1 central PLC with 2 touch panels, 2 FL-net networks, and 1 VME computer.

Local PLC Units

We simply replaced one embedded PC and its corresponding DICMs with a new local PLC because of the time and space limitations for our upgrade. We adopted a MELSEC Q-series controller [4], equipped with a 12-inch V812iS touch panel with 800 × 600 resolution as the local PLC [5]. Figure 3 shows the local PLC unit mounted on the existing 19-inch rack.

The local PLC uses an RS-232C module to interface with the optical converters that use serial communication with a different signal level than the local PLC. The local PLC is equipped with several two-channel RS-232C modules and communicates with the optical converter through a newly prepared external circuit that converts the signal level to that of the optical converter.

We give integration and evaluation roles for the 1-h integrated dose and the newly introduced 1-week integrated dose to the local PLC instead of the existing PLCs. The local PLC outputs interlock signals to ARSIS when the following conditions are satisfied.

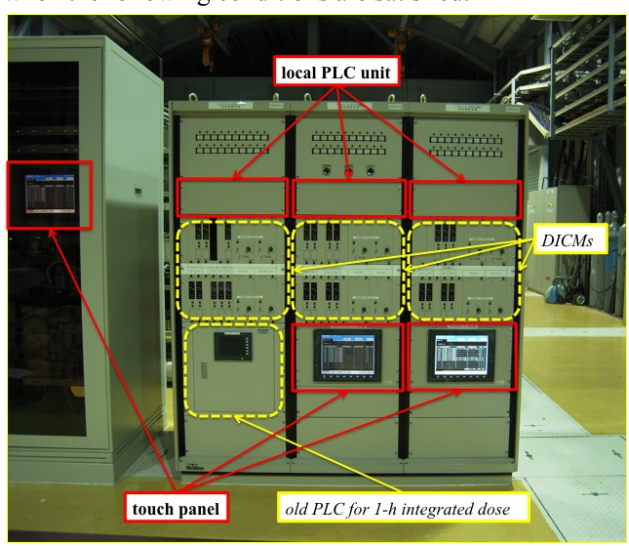


Figure 3: Photograph of the local PLC unit.

- The radiation value is lower than the low threshold level during the 3 s when the interlock by low alert is enabled.
- The radiation value is higher than the high-high threshold level during the 3 s when the interlock by high-high alert is enabled.
- The 1-h integrated dose becomes larger than its threshold level when the interlock by 1-h integrated dose is enabled.
- The 1-week integrated dose becomes larger than its threshold level when the interlock by 1-week integrated dose is enabled.
- The radiation monitor or the communication device encounters an error when the interlock by system failure is enabled.

These thresholds can be changed using the local touch panels. Alert histories encountered on the local PLC can be viewed on the touch panel.

We have incorporated the data-logging function into the local PLC unit. An 8-GB compact flash drive installed into the local touch panel stores sampling data for 7 days. This function ensures continuous data logging even when data acquisition into the accelerator database is stopped for some reasons.

Central PLC Unit

The new central PLC unit has been deployed in the radiation monitoring room near the central control room to enable centralized management of all the radiation monitors. The central PLC unit is equipped with two FL-net interface modules. One module communicates with the twelve local PLCs, while the other module communicates with the VME computer. We have also adopted the MELSEC Q-series PLC with two touch panels.

The central PLC unit simply oversees the entire system management; that is, it does not play critical roles related with ARSIS. The central PLC unit provides the following functions.

- Browse the current radiation data and status of all the radiation monitors.
- Browse 1-h and 1-week integrated dose data for the interlock monitors.
- Browse the alert histories occurring in all the radiation monitors.
- Change the alert/warning thresholds of low/high/high-high/1-h integrated dose/1-week integrated dose of all the radiation monitors.
- Enable/disable each interlock function of high-high/1-h integrated dose/1-week integrated dose.
- Attach/detach the radiation monitors to/from the surveillance system.
- Change the system times of all the PLC units.
- Change the common password to enable privileged accesses to the all the PLC units.

To enable the VME computer to collect the data, the central PLC copies almost all the data from the local PLC on the FL-net for the VME computers.

UPGRADE

We faced several difficulties during upgrade. The biggest problem was insufficient time for upgrade. Although the work had to be completed by 18th March 2011, it started after the SPring-8 operation was stopped in the end of February. Furthermore, we were not able to start the detailed designing work with the contractor until November 2010, even though we had already finished the ground design.

The restriction that the old devices such as the DICMs had to be kept in 19-inch racks made the work troublesome. On the other hand, this restriction enabled the old system to be restored immediately even when the new system did not work well. However, there was not enough space left to install the new system in the existing 19-inch racks.

In order to overcome these difficulties, we carried out the upgrade following a well-defined process.

- We prepared a ladder program and a level conversion circuit for the communication test held in the winter shutdown period of 3 weeks that was the unique opportunity for the test before the end of the February. The preparation of the ladder program and the level conversion circuit was carried out according to the specifications.
- During the winter shutdown, we investigated the communication protocols between the DICMs and the radiation monitors, and we tested the prepared program and circuit with the actual system.
- After the winter shutdown, we prioritized the design of touch panels because they made clear the necessary functions of both local and central PLCs. We discussed the design with the safety office, the main user of the new system.
- The PLC units were designed simultaneously. The units had to be as compact as possible because of shortage of installation space in the 19-inch racks. All the level conversion circuits and terminal blocks for external signals were put together in the units.

Consequently, we successfully upgraded the system before the deadline. The remaining minor troubles were fixed during a short shutdown period from 28th April 2011 to 9th May 2011.

SUMMARY AND FUTURE PLAN

We have re-designed the data acquisition system for the SPring-8 radiation monitors. The new system is a PLC-based system and is simpler, more comprehensible, and is more reliable than to the previous system. The new system consists of 12 local PLCs and 1 central PLC. The introduction of the central PLC enables the centralized monitoring and management of all the radiation monitors. All the local and central PLCs are equipped with touch panels to facilitate ease in operation for the radiation monitors.

The new system has been employed in the practical use since the end of March 2011 and has been working well

without any issues after a minor problem was fixed in May 2011.

In August 2011, we succeeded in adding the same setup functions to the radiation monitors as those employed in the DICMs. These functions are mainly used for maintenance work during long shutdown intervals. The new system was successfully applied for the practical maintenance work, in August 2011. This proved that the new system was realistic enough for maintenance work. This will enable us to eliminate DICMs from the radiation monitoring system and rearrange the 19-inch racks to enhance maintainability by eliminating the dispensable devices.

REFERENCES

[1] C. Saji, et al., "Upgrade of Accelerator Radiation Safety System for SPring-8", Proc. of ICALEPCS'09, Kobe, Japan, 2009

[2] R. Tanaka, et al., "The First Operation of Control System at the SPring-8 Storage Ring", Proc. of ICALEPCS'97, Beijing, China, 1997, p. 1.

[3] http://www.jema-net.or.jp/English/standard/opcn_e/opcn05.htm

[4] http://www.mitsubishielectric.com

[5] http://www.hakko-elec.co.jp/index_e.html

STATUS, RECENT DEVELOPMENTS AND PERSPECTIVE OF TINE-POWERED VIDEO SYSTEM, RELEASE 3

Stefan Weisse, David Melkumyan (DESY, Zeuthen, Germany),

Philip Duval (DESY, Hamburg, Germany)

Abstract

Experience has shown that imaging software and hardware installations at accelerator facilities needs to be changed, adapted and updated on a semi-permanent basis. On this premise the component-based core architecture of Video System 3 was founded. In design and implementation, emphasis was, is, and will be put on flexibility, performance, low latency, modularity, interoperability, use of open source, ease of use as well as reuse, good documentation and multi-platform capability. In the past year, a milestone was reached as Video System 3 entered production-level at PITZ, Hasylab and PETRA III. Since then, the development path has been more strongly influenced by production-level experience and customer feedback. In this contribution, we describe the current status, layout, recent developments and perspective of the Video System. Focus will be put on integration of recording and playback of video sequences to Archive/DAQ, a standalone installation of the Video System on a notebook as well as experiences running on Windows 7-64bit. In addition, new client-side multi-platform GUI/application developments using Java are about to hit the surface. Last but not least it must be mentioned that although the implementation of Release 3 is integrated into the TINE control system [1], it is modular enough so that integration into other control systems can be considered.

OVERVIEW

The Video System explained further on originates at the Photo Injector Test Facility Zeuthen (PITZ). This is a test facility at DESY for research and development on laser driven electron sources for Free Electron Lasers (FEL) and linear colliders [2, 3]. A vital diagnostic used to observe and measure the electron beam is the Video System at PITZ. As the facility has been constantly upgraded over the last decade, the video system was enlarged and easily outgrew its initial specifications.

Based on the acceptance and extension at PITZ, the Video System has been exported over the years to PETRA at DESY and its pre-accelerators [4], EMBL Hamburg and PETRA III user beamlines at Hasylab [5], where the requirements for a Video System at these facilities admittedly were substantially different compared to PITZ. In addition, remarkably small installations have been realised as well, such as providing video stream of a high-voltage power supply display installed at a remote location.

During the years of evolvement, the Video System underwent constant extension and upgrading. Naturally, it quickly outgrew its original specifications which made any further extension or upgrading cumbersome. A redesign towards a component-based approach, where individual elements are easy to change, adapt, upgrade etc., has been completed [6]. Due to this well-defined and flexible approach, a lifetime of more than a decade is expected. This fits to lifetimes of existing and coming accelerators and other experiments close to basic physics research.

Some of the many new available features include:

- flexible image source interface, focused to AVT Prosilica and JAI Gigabit Ethernet cameras, other interfaces available, new interfaces easy to implement
- raw grayscale images up to 16 bits per pixel
- raw colour images (RGB24)
- integrated JPEG compression/decompression (grey and colour)
- focused but not limited to Gigabit Ethernet, GigE Vision, gen<i>cam cameras
- loss-less acquisition, transport and analysis
- high-bandwidth (tens of megabytes per second)
- low latency (tens of milliseconds range)
- well-defined image data type, integrated in TINE data transport
- up to 30 frames per second are easily achievable
- lightweight shared memory interconnection for video stream handover across video system components
- multicasting of video images via TINE
- interface to Matlab
- well-defined, documented and flexible XML configuration files
- integrated native Java-based universal slow control solution (puts up a pretty good show on its own)
- native components (MS Windows) are based on platform-independent source code easy to port
- client-side is native Java code

It needs to be mentioned here that the system has been kept very flexible beneath the surface. Use-cases that require more frames per second, integration of other cameras, higher resolutions or bit-depths than mentioned above can certainly be considered. If there is demand, a test installation to verify special requirements should be easy to provide.

STATUS

At the moment, the current release of the Video System has been installed and is used in production level at PITZ, Petra III (pre-accelerators, diagnostic beamline, synchrotron radiation interferometer, user-beamlines of Hasylab, …) and since April 2011 at the Relativistic Electron Gun for Atomic Exploration (REGAE) accelerator. In the past, the European Molecular Biology Laboratory (EMBL)

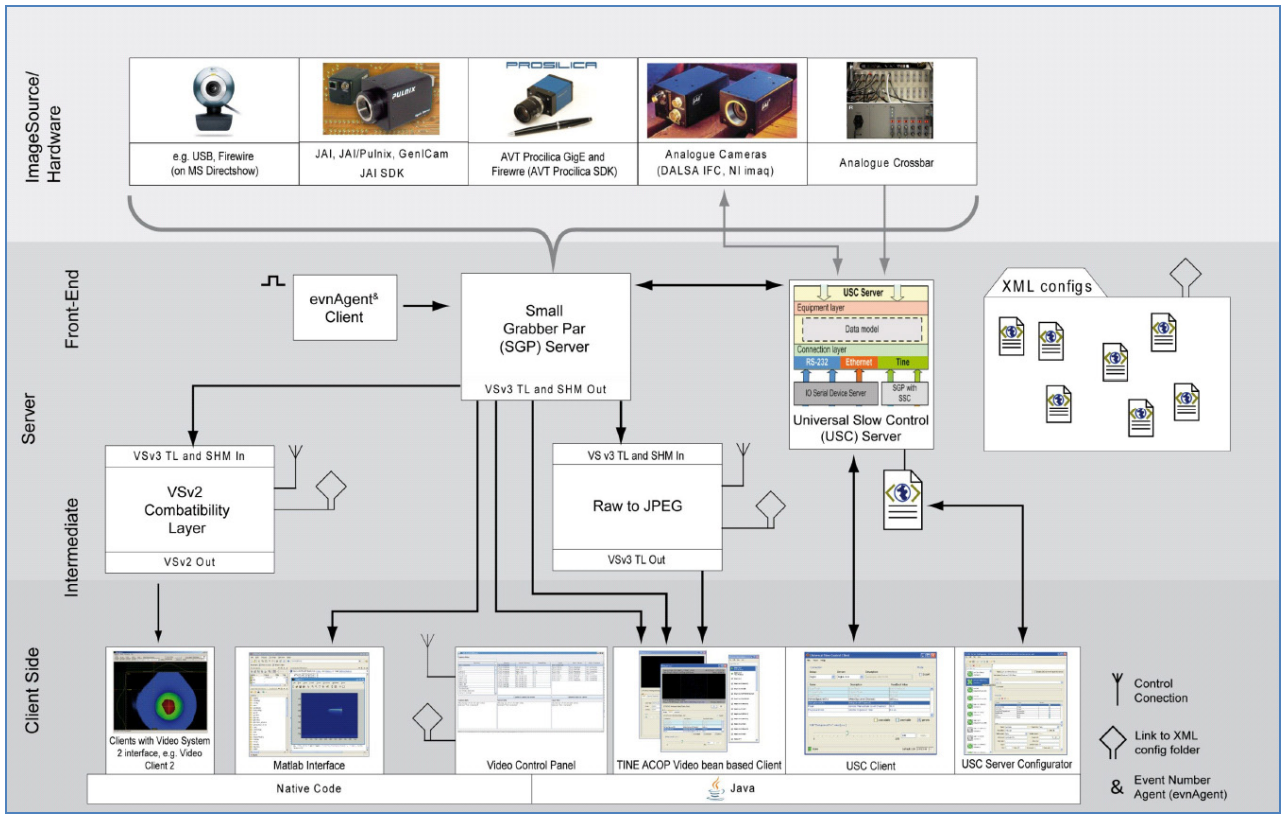


Figure 1: Simplified layout of VSv3 components and their interaction.

Hamburg has used the Video System 2 (VSv2) [7] for sample changer monitoring and control to great satisfaction. As step by step EMBL user beam-lines are commissioned at PETRA III, VSv3 components are foreseen to be installed there. The simplified layout of the VSv3 components can be seen in Fig 1.

PITZ

In the past year, PITZ has finished its current gun conditioning cycle and is currently experiencing a hardware upgrade of the accelerator. The video system is set up in a heterogeneous way. Various types of cameras (analogue JAI, digital JAI, JAI/Pulnix and Prosilica GigE) coexist next to each other, each one where its characteristic fits best. All cameras are snapshot-triggered by central timing (mainly 10 Hz) and are in general delivering frames at this rate. Legacy VSv2 technology is still used on the client side, while back-end/infrastructure has been fully upgraded to the current release level. Matlab interfaces keep growing as to satisfy demands of physicists.

PETRA III

Upgrading VSv2 legacy installations to VSv3 was one of the main tasks performed during the past year. Now as everything is VSv3-based, a Raw to JPEG component has been introduced almost anywhere to reduce video transmission usage on the controls network. For many raw video feeds to the controls network there is now an additional JPEG feed which provides the same stream, only JPEG-compressed (using compression ratios of about 1:10 to 1:50). In addition, a synchrotron radiation interferometer has been commissioned and the diagnostics beam-line has been upgraded to JAI BM141 camera. Matlab interfaces have been introduced and extended in the last months. The synchrotron radiation interferometer measurement applet is based on this.

PETRA III client-side at Hasylab

In the past year, the already existing pure VSv3 installation at client-side of Petra III has constantly been extended. Currently there are various types of Prosilica GigE cameras (about 90 pieces) running in parallel at slow non-triggered update rates. As many server processes as cameras, distributed across three server hosts running Windows XP SP3, provide control of the cameras and a standard VSv3 video interface via TINE to the control system.

REGAE

A new linear accelerator is currently being set-up at DESY. For the diagnostics of the electron bunches at REGAE [8], an installation of the Video System has been done. For initial operation phase, three cameras of JAI BM 141 have been installed at the accelerator. One server

PC is used for control and readout of these cameras. First beam images have been taken.

RECENT DEVELOPMENTS

Transition from former technology (Video System 2) to the state-of-the-art Video System is still ongoing. At PITZ, step by step important legacy client applications such as the PITZ Video Control Panel and Video Client 2 are going to be exchanged to new native Java-based applications.

Foreseen as a successor to Video Client 2, the Java Video Client application is a component-based client-side application, which encapsulates the functionality needed to provide a single high-level user interface to display video streams including image analysis (ACOP Video component) and controlling camera settings (USC component) as well as providing a consistent view of all available cameras across a VSv3 installation.

Matlab Interface

In 2010, the first interface from VSv3 to Matlab was created and used enthusiastically by the physicists, who immediately began requesting more features. Thus, as time passed, the Matlab interface grew. Using the multi-platform-available MEX-functions, one is able to get a single video image from the server, load in proprietary Video System 2 images and image sequences from file and is able to grab a sequence of video images from server. The MEX functions have been carefully designed for cross-platform compatibility and have been made available on a wide range of Matlab versions (v7 to 2011a) on Win 32, Linux and Win 64. Our aim here is again to provide the user with a wide range of tools, so that he can use his favourite working environment. As our development cycle regarding the Matlab interfaces is maturing and the physicists are providing feedback, more external functions can be expected to be implemented.

Video System on a Notebook

In the last year demand rose for having a standalone Video System installation on a notebook, where one is able to use the benefits of a known on-site video system installation for both on- and off-site use. This allows temporary local installations at any laboratory, where hands-on experience can be gained by sites interested in joining the video system collaboration For example, the Cherenkov Telescope Array (CTA) test facility to be built in Berlin is interested in having such a local installation on a notebook for first hands-on outdoor measurements on and at the dish.

Some local installations on notebooks have been used, primarily at REGAE and CTA. Each has in common a server as well as client applications which can be run on the notebook locally (see Fig 2). A local Ethernet interface is used to connect a GigE camera. As a bonus, the software was set up in such way that if a network connection to a site-network exists, video system components located therein can be accessed if necessary. As Windows XP SP3 Professional workstation-type machines are used for video server environment on site, installation on Windows XP notebook was fairly straightforward. In contrast, a CTA notebook was already equipped with Windows 7 / 64 bit. Although this was new territory, the installation and usage of native Windows (Win32) software under 7/64 was easier than expected. However, significant attention needed to be paid to the Start Menu and shortcuts, WOW64 and Networking Firewall.

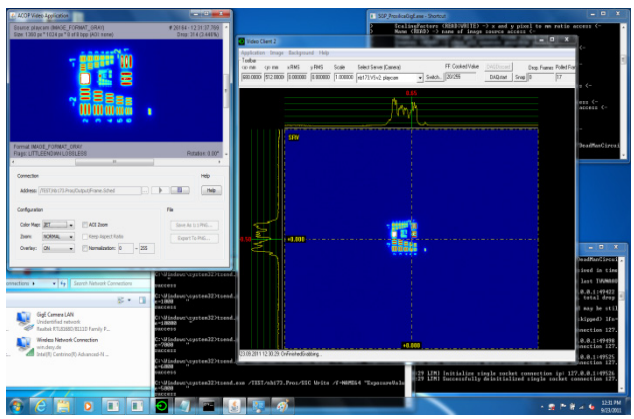


Figure 2: Video System 3 and 2 (client-side) running in parallel on a notebook using Windows 7/64 bit.

Attaching Event Numbers to Video Frames

Up to now, a neglected part of the Video System has been the integration to Archive/DAQ systems. As an important step in this direction, attaching event numbers to newly acquired video frames has been implemented (see Fig 3). An important criterion for exact correlation of a beam image at a facility using triggered acquisition like PITZ is a so-called general event number. Each shot of the accelerator gets a unique event number assigned, which in general is transported to individual controls subsystems and experiment's properties are tagged with that event number. Afterwards, the data is transferred to the DAQ system. Later on, the physicist will require the event number to collect data that belong to an individual shot of the accelerator. A lightweight solution has been developed and installed at PITZ, which makes use of multicasting the event number from a central timing source throughout the controls network. On the server level, the SGP component with the help of an 'event number Agent' component (evnAgent), tags each freshly acquired video image with the event number.

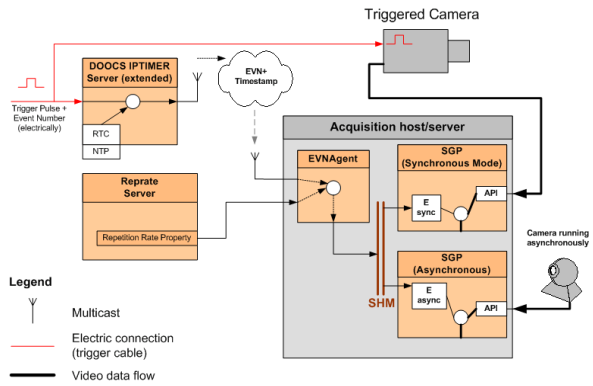


Figure 3: Distribution of Event Number.

The integration of image data into the TINE event archive system is well underway. Generic tools such as the TINE Event Archive Viewer are now able to store and retrieve video sequences, either frame-by-frame or movie style. Streaming these stored image data into specific dedicated video analysis tools will provide the physicist with the ultimate comfort in rapid browsing and searching of stored events coupled with the precision analysis found in the live video tools.

Online Documentation

The Video System website (an important element of the TINE website) has likewise matured over the past year [9]. General information, technical aspects, reference documentation, conference submissions and contact persons are now available world-wide. Readers are kindly asked to provide feedback.

PERSPECTIVE

At REGAE, integration of Andor iXon scientific detector into the Video System is currently being considered. By using a plain C-API which Andor provides to interface with the detector, a proof-of-concept application has been successfully implemented. A SGP component may be introduced later on.

A big point for PITZ in the coming year is the removal of Video System 2 support on the client side. Necessary for this is the availability of comparable tools using Video System 3 interface. Most tools will follow the paradigm of using native Java on the client side, to provide freedom of client platform for the operator. Also, effort will be spent to move legacy native measuring clients with VSv2 interface to the new release level. The Emittance Measurement Wizard (EMWiz) and Momentum and Momentum Analysis tool (MAMA) are two central applications there.

Furthermore, investigations and implementations of native releases of some Video System components on Windows 7/64 bit und possibly Linux platform, implementation of Image Source Profiles, creation and extensions of client VSv3 APIs, fine tuning of server-side to make it easier to set-up, maintain and use, are certainly worth mentioning.

ACKNOWLEDGEMENTS

We would like to thank Kathrin Pflaum, Markus Degenhardt and Peter Walter at Hasylab for administration and growth of the Video System at their site, Gero Kube for substantial interest and feedback, providing of sample cameras and beta-testing and Hossein Delsim-Hashemi for pushing things forward at Regae site.

REFERENCES

[1] http://tine.desy.de
[2] F. Stephan, C.H. Boulware, M. Krasilnikov, J. Baehr et al., “Detailed characterization of electron sources yielding first demonstration of European X-ray Free-Electron Laser beam quality”, PRST-AB, Vol. 13, No. 020704 (2010)
[3] S. Rimjaem et al., “Measurements of Transverse Projected Emittance for Different Bunch Charges at PITZ”, FEL 2010, Malmö, Sweden
[4] G. Kube et al., “Petra III Diagnostics Beamline for Emittance Measurements”, IPAC 2010, Kyoto, Japan
[5] M. Degenhardt et al., “CVD Diamond Laser Alignment and X-Ray Fluorescent Screens for Petra III”, SNI 2010, Berlin, Germany
[6] S. Weisse, D. Melkumyan, P. Duval, “Status, Applicability and Perspective of TINE-powered Video System, Release 3”, PCaPAC 2010, Saskatoon, Canada
[7] S. Weisse et al., “Status of a versatile Video System at PITZ, DESY-2 and EMBL Hamburg”, ICALEPCS 2007, Knoxville, TN, USA
[8] Sh. Bayesteh, H. Delsim-Hashemi. “Diagnostics of Femtosecond Low-Charge Electron Bunches at Regae”, IPAC 2011, San Sebastian, Spain
[9] http://adweb.desy.de/mcs/tine/VideoSystem

SOFTWARE RENOVATION OF THE CERN EXPERIMENTAL AREAS

J. Fullerton, L. Jensen, J. Spanggaard, CERN, Geneva, Switzerland

Abstract

The Experimental Areas at CERN's Antiproton Decelerator (AD), Proton Synchrotron (PS) and Super Proton Synchrotron (SPS) have recently undergone a wide-spread electronics and software consolidation based on modern techniques. This paper will describe the scale of the software renovation and how the issues were overcome in order to ensure a complete integration into their respective control systems.

INTRODUCTION

The CERN Experimental Areas include 7km of beam lines (Figure 1) with just over 400 detectors used by about 2400 experimental physicist and covering three different locations: the North Area (SPS), the East Area (PS) and the Antiproton Decelerator (AD). This instrumentation allows the observation of particle beams via 7 families of detectors (Table 1). The main requirement for the consolidation was to reduce down-time for physics and to reduce the required man-power for CERN to maintain and operate the beam instrumentation in these lines.

Table 1: Detector Families

Detector family	Number in the Areas
Scintillation Counters [1]	50
Filament Scintillators	65
Cerenkov counters	8
Delay wire chambers[2]	36
Analog wire chambers	65
Gas electron multipliers[3,4]	4
Scaler-counters.	190

HARDWARE SOLUTION

Until recently the hardware to control the beam instrumentation was implemented with CAMAC and NIM modules that had become obsolete and were generating frequent faults. In order to make the hardware more reliable and simpler to troubleshoot, new equipment-oriented hardware was developed in VME [1]. About 30 such VME crates are now controlling all the beam instrumentation equipment in CERN's experimental areas.

Figure 1: A beam line in the North Experimental Area.

SOFTWARE SOLUTION

The software structure chosen to accompany this hardware renovation is based on a classic three tier structure (Figure 2). The operational user GUI connects to the **C**ERN **E**xperimental area **S**oftw**A**re **R**enovation (CESAR) [5] J2EE server from where the Controls Middle Ware (CMW) [6] grants access to the Front End Software Architecture (FESA) server [7] and configuration databases.

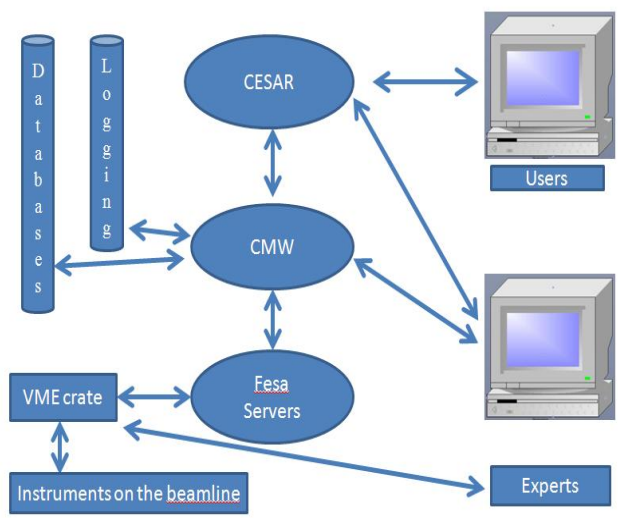


Figure 2: The three tier Software structure used.

CESAR

This tier allows a high level of abstraction. From the controls point of view the VME crates access detectors on many different beam lines whereas the end-user only wants to see equipment on the beam line on which he/she is working. This extra level of abstraction enables all the access rights, beam line configuration and display code to be separated from the hardware server code.

CMW

An object oriented device model implementation middleware with extensions to cover accelerator control created for the demands of LHC.

FESA

An environment for developing, deploying and maintaining front-end software. This enforces CERN accelerator control standards.

The equipment specialist can access hardware on four levels:

- CESAR GUI's where they have the same functionality as the users, which often allows simple misunderstandings to be ironed out.
- Expert GUI's (Figure 3) providing extra, potentially dangerous functionality, where processes such as pressure scans can be carried out. A separate expert GUI shows the status of all equipment of a single type (shown in Figure 4), which is useful as the equipment specialist often tests by instrument type and not by beam-line.
- FESA navigator which is a data driven GUI program for setting and getting all parameters defined in the front-end FESA class.
- Locally run programs on the VME front-end with no complex communications software involved.

These separate possibilities are important as they allow each level of the structure to be tested separately in case of doubts whether an issue is related to hardware or a software tier.

All computers, VME crates and the various servers running on them are constantly monitored by the CERN Controls Group "DIAMON" tool, a universal tool for diagnostics on the controls infrastructure across all CERN accelerators.

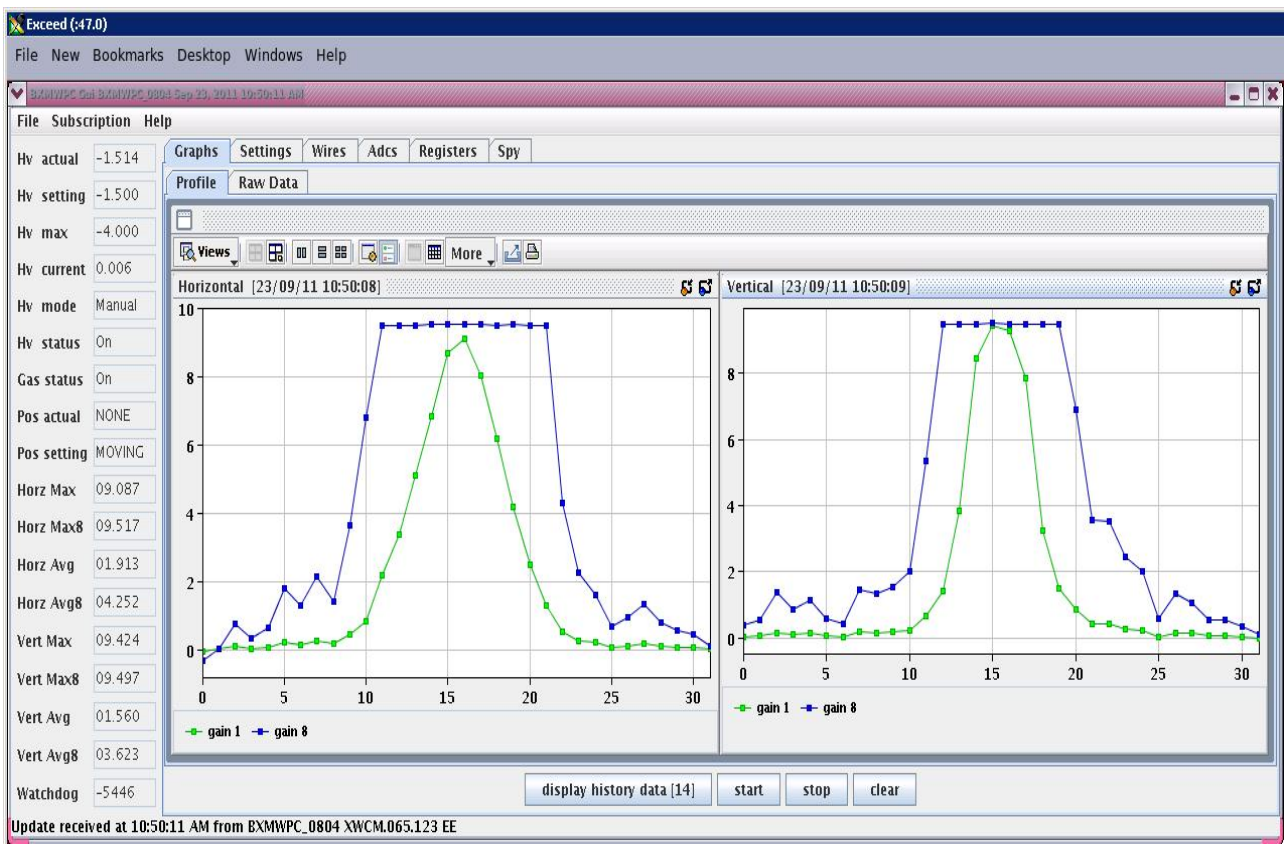
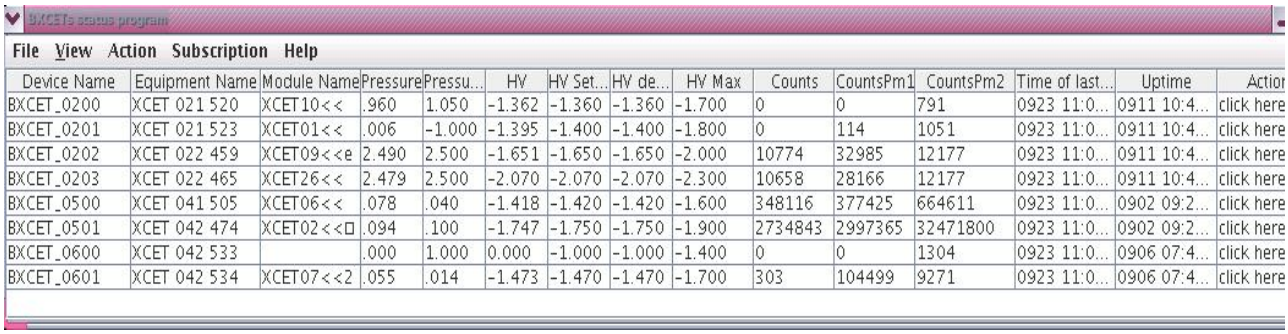


Figure 3: Example of an expert GUI, this one is for the control of an Analog wire chamber.



Device Name	Equipment Name	Module Name	Pressure	Pressu...	HV	HV Set...	HV de...	HV Max	Counts	CountsPm1	CountsPm2	Time of last...	Uptime	Actior
BXCET_0200	XCET 021 520	XCET10<<	.960	1.050	-1.362	-1.360	-1.360	-1.700	0	0	791	0923 11:0...	0911 10:4...	click here
BXCET_0201	XCET 021 523	XCET01<<	.006	-1.000	-1.395	-1.400	-1.400	-1.800	0	114	1051	0923 11:0...	0911 10:4...	click here
BXCET_0202	XCET 022 459	XCET09<<e	2.490	2.500	-1.651	-1.650	-1.650	-2.000	10774	32985	12177	0923 11:0...	0911 10:4...	click here
BXCET_0203	XCET 022 465	XCET26<<	2.479	2.500	-2.070	-2.070	-2.070	-2.300	10658	28166	12177	0923 11:0...	0911 10:4...	click here
BXCET_0500	XCET 041 505	XCET06<<	.078	.040	-1.418	-1.420	-1.420	-1.600	348116	377425	664611	0923 11:0...	0902 09:2...	click here
BXCET_0501	XCET 042 474	XCET02<<□	.094	.100	-1.747	-1.750	-1.750	-1.900	2734843	2997365	32471800	0923 11:0...	0902 09:2...	click here
BXCET_0600	XCET 042 533		.000	1.000	0.000	-1.000	-1.000	-1.400	0	0	1304	0923 11:0...	0906 07:4...	click here
BXCET_0601	XCET 042 534	XCET07<<2	.055	.014	-1.473	-1.470	-1.470	-1.700	303	104499	9271	0923 11:0...	0906 07:4...	click here

Figure 4: A display of all the Cerenkov counters of the North Area.

STATUS

FESA Servers and Expert GUI`s have been developed for all new equipment-oriented hardware modules and we are now in the process of introducing new detector families, requiring similar solutions. In parallel existing hardware modules are being used to control new instrument-types with only minor changes to the servers. An example of this is for the East Area target telescopes that will benefit from the existing VME module and server of the Scintillation Counter. The target telescope is an intensity measuring device based on scintillation that observes the particle production through a 5 meter long pipe perpendicular to the beam axis.

VME front-end crate controllers are currently changing from LynxOS to Linux. This move was in parts caused by modern software being memory hungry. The increase in speed of the new controllers has made the need for a specific real-time operating system an unnecessary overhead. The switch of operating system and controller card has used more manpower than expected due to the requirements/exploitation of current standards e.g. C to ANSI C and missing features in the new compilers which were used unknowingly in our old code.

CONCLUSION

As a result of this hardware and software renovation CERN's Beam Instrumentation Group has been able to support more experimental users, with fewer CERN staff dedicated to these areas. The instrumentation of the experimental areas is now based on modern, maintainable software tools and on the hardware side the new equipment-oriented approach has increased the reliability of the electronics. Both of these consolidations have made the overall maintenance load significantly smaller. From a user point of view the control system has changed from a command line to a more intuitive GUI based interface that provides better displays. A further side effect of using CERN standard software solutions has been to enable other software experts to remedy issues where formerly only the author of the code had the needed knowhow.

REFERENCES

[1] G. Baribaud, C. Beugnet, A. Cojan, G.P. Ferri, J. Fullerton, A. Manarin, J. Spanggaard "Control modules for Scintillation Counters in the SPS Experimental Areas" DIPAC 2001

[2] CERN Technical Note SL-Note-98-023 (BI) Delay Wire Chambers, a users guide

[3] Jacques Bosser, Kondo Gnanvo, Jens Spanggaard, Gerard Tranquille (CERN) New Beam Profile Monitor Based On GEM Detector for the AD transfer and experimental lines EPAC 2004

[4] J. Spanggaard, F. Arnold, P. Carriere, L. Ropelewski, G. Tranquille, CERN, "Gas Electron Multipliers for low energy beams" IPAC10

[5] V. Baggiolini, P. Bailly, B. Chauchaix, F. Follin, J. Fullerton,Ph. Malacarne, L. Mateos-Miret, L. Pereira "The Cesar Project – using J2ee for accelerator controls " ICALEPCS 2003

[6] K. Kostro, V. Baggiolini, F. Calderini, F. Chevrier, S. Jensen, R. Swoboda, CERN, Geneva,N. Trofimov, IHEP, Protvino, Moscow Region "Controls Middleware - The New Generation" EPAC 2002

[7] Michel Arruat, Leandro Fernandez, Stephen Jackson, Frank Locci, Jean-Luc Nougaret,Maciej Peryt, Anastasiya Radeva, Maciej Sobczak, Marc Vanden Eynden Accelerators and Beams Department, CERN, Geneva, Switzerland "**F**ront-**E**nd **S**oftware **A**rchitecture" ICALEPCS 2007

A BEAM PROFILER AND EMITTANCE METER FOR THE SPES PROJECT AT INFN-LNL

G. Bassato, A. Andrighetto, N. Conforto, M. Giacchini, J. A.Montano, M. Poggi, J.A. Vásquez
INFN-LNL, Legnaro (PD), Italy

Abstract

New beam diagnostics tools have been developed for the SPES [1] project in the perspective of reusing them to upgrade the system currently in operation at LNL in the superconducting Linac ALPI.

The goal is providing the new R.I.B. facility, that will use ALPI as re-accelerator, with an homogenous set of tools and a common user interface to support beam transport over the future accelerators complex. This paper focuses on the realization of a new emittance meter and on design of control software that has been written using EPICS.

OVERVIEW

The Spes Project and the New Ion Source.

SPES is a new facility under construction at LNL whose aim is the production of radioactive nuclei beams that will be injected into the Linac ALPI. The production of unstable nuclei is accomplished by impinging a proton beam delivered by a commercial cyclotron (energy range 40-70 Mev) into a UCx target. The diagnostics on proton beam will be part of cyclotron delivery and will not be presented in this paper; instead, we shall discuss the diagnostics on secondary beam with particular emphasis on devices that have been realized to characterize the new ion source currently under test in the SPES Target laboratory.

The Beam Diagnostics at LNL

The Linac ALPI, together with its injector PIAVE, has about 80 QWR resonators in operation. Beam transport is carried out by observing its profile and measuring its current by means of 48 diagnostic boxes installed along the machine. Each diagnostic box contains a faraday cup and a couple of grids (horizontal and vertical), each one made of 40 wires. Current signals are converted to voltage by an analogue board closely installed on top of box, in order to minimize the noise induced by cables; the conversion factor is selectable by software and covers four decades. Grid signals are multiplexed and serialized before being transferred, over coaxial cables, to the data acquisition system. Serialization strongly simplifies the overall wiring: in fact, only one coaxial cable is required for a 40 wire grid The multiplexer is driven by a counter whose clock is generated by the ADC itself, to have the signal transmission synchronized with the conversion.

DEVICES FOR BEAM DIAGNOSTICS

Figure 1 shows the faraday cup designed for beam current measurement. Behind this device it is possible to notice the grid based profiler. The faraday cup is inserted or removed by means of a pneumatic actuator while the grids are moved by a stepper motor.

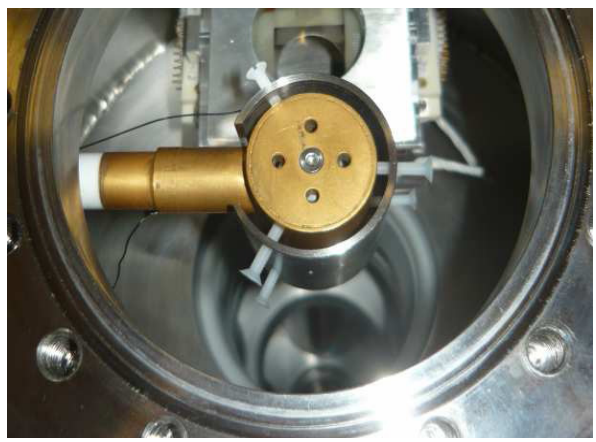


Figure 1: Faraday cup and beam profiler.

The Data Acquisition System

Signals from beam profilers and faraday cups are acquired by ADC cards installed in VME crates. The boards are XVME566 produced by XYCOM; they have 12 bit resolution and support a conversion rate of 100 KHz in streaming mode. Although higher performance products are available today we decided to continue using these modules, already installed in ALPI diagnostics system, because of the on board timer chip we use to generate the clock for the external multiplexer.

The stepper motor controllers are VME cards, designed in house, together with the associated power drivers; one controller manages 8 motors. The crate controller is an Emerson MVME3100, based on the PowerPC8540 processor and including 256MB RAM. The need of keeping the stream of serialized analogue signals coming from front-end electronics synchronized with the A/D conversion cycle imposes strict constraints in terms of response time. Because of this, the data acquisition software runs under the real time operating system Vxworks; the release currently in use is 6.8. The OS image has been tailored to support all features required to host an EPICS IOC using Wind River Workbench 3.2.

THE EMITTANCE METER

Measuring the emittance is of essential importance to evaluate the quality of beam produced by an ion source. The emittance meter designed for SPES [2] is based on two identical moveable slits (collimators) placed in front of a couple of horizontal and vertical grids. The slits have an aperture of 0.3 mm. and the distance from grids is 300 mm. By moving the collimators up and down (or right to left and acquiring data from vertical grids) it is possible to scan the whole beam area and evaluate the beam divergence by measuring the grid currents for different collimator's positions (the position is obtained by reading the output voltage of a linear encoder). Figure 2 illustrates the measurement scheme, while figure 3 shows the mechanical realization.

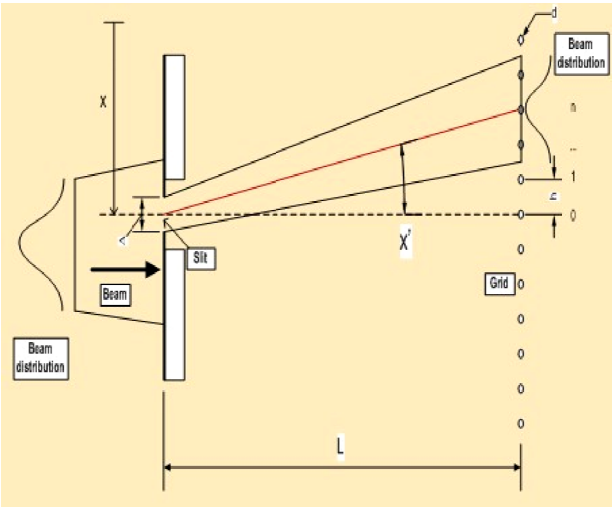


Figure 2: Scheme of emittance measurement.

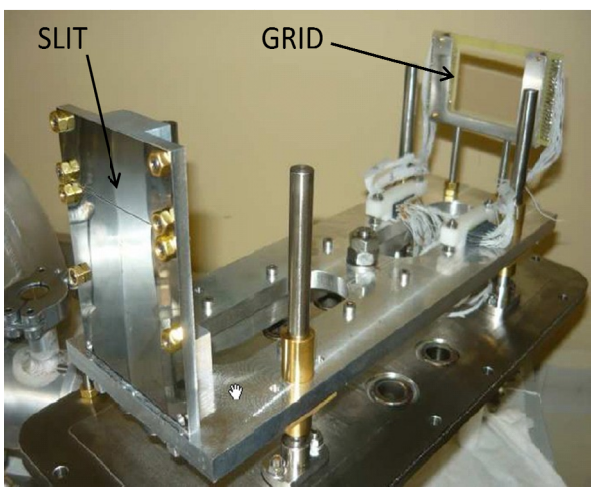


Figure 3: Instrument mechanics.

SOFTWARE DEVELOPMENT

One key point of SPES new diagnostics system – and in perspective of ALPI diagnostics renovation – is the complete remake of control software. The choice of software tools for beam diagnostics development was coherent with our previous decision of using EPICS as general framework for SPES controls. It will allow us to take advantage of a plenty of utilities developed by the user community other than the great performance of data distribution system offered by Channel Access architecture.

This choice also fits naturally with our plan of ALPI diagnostics upgrade: reusing the rather large amount of legacy hardware installed in ALPI will be not a problem since the XYCOM I/O boards and the operating system Vxworks are supported by EPICS since its origin. Some modifications were required on drivers available from the community but the overall cost of migration will be strongly reduced.

The Emittance Measurement Software

According to EPICS terminology, the VME processor implements an IOC (Input Output Controller) providing the following functions:

- acquisition of grid signals, an array of integers configured as an instance of waveform record
- acquisition of collimator position from the linear encoder
- control of stepper motors, to synchronize the collimator motion with the grid signals acquisition

The database loaded in the IOC is relatively simple and provides minimal processing on raw data, mainly using C-sub records. Most calculations are performed on the client side, which is a Linux PC also used to run the graphic interface. The program is written in C++; it reads through the Channel Access the grid current arrays acquired by subsequent samples together with collimator position and stores all information in its disk, until the number of programmed steps is reached. At the end of the acquisition cycle it calculates the emittance and displays the result.

Data used for emittance analysis can also be stored in an EPICS Archiver for off-line review. The Archiver version currently in use is configured to work in conjunction with mySQL running on a Linux server. Replacement by a higher performance installation based on Hypertable [3] is planned for the near future.

The user interface has been realized using CSS (Control System Studio)[4], the new Java based graphic environment that provides a seamless integration of EPICS records management and high level supervisor functions (i.e. alarm handling, interface to the Archiver, etc.). Figure 4 is a snapshot of user interface, showing in a unique panel the raw beam profile as collected by grids, the faraday cup current and the phase space distribution.

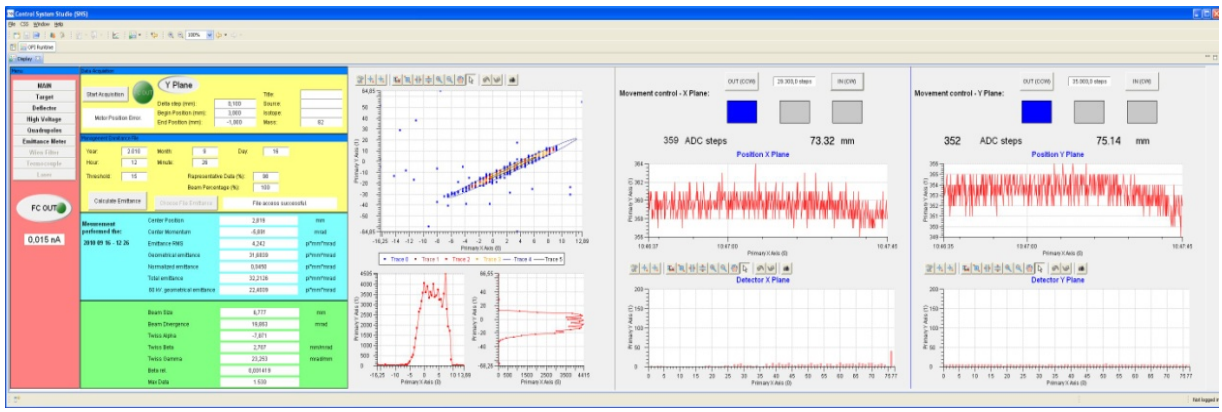


Figure 4: Snapshot of user interface based on CSS.

CONCLUSION

New diagnostics instruments were developed for SPES and are today in use in the Ion Source and Target Laboratory. The emittance meter has proven to be a very sensitive tool for the characterization of beams generated by the ion source prototype. The software migration to EPICS resulted in a significant increase of flexibility and performance of overall system. The software developed for SPES is being ported with minimal adaption to ALPI, in order to have an unique interface for the diagnostic systems of both facilities.

ACKNOWLEDGMENTS

We are grateful for their valuable help to M. Davidsaver (BNL) for rewriting the XYCOM566 driver and to D. Chabot (BNL) for developing the EPICS driver for our stepper motor controller.

REFERENCES

[1] http://www.lnl.infn.it/~spesweb/
[2] J. Montano, J. Vasquez et al. “Off line emittance measurements of the SPES ion source at LNL”, N.I.M.-A, vol. 648, aug. 2011
[3] http://www.lnl.infn.it/~epics/archiver.html
[4] http:// ics-web.sns.ornl.gov/css/

UPGRADE OF THE NUCLOTRON EXTRACTED BEAM DIAGNOSTIC SUBSYSTEM.

Evgeny V. Gorbachev, Nicolay Lebedev, Nicolay Pilyar, Sergey Romanov, Tatyana Vladimirovna Rukoyatkina, Valery Volkov, JINR, Dubna, Russia

Abstract

The subsystem is intended for the Nuclotron extracted beam parameters measurements. Multiwire proportional chambers are used for transversal beam profiles measurements in four points of the beam transfer line. Gas amplification values are tuned by high voltage power supplies adjustments. The extracted beam intensity is measured by means of ionization chamber, variable gain current amplifier DDPCA-300 and voltage-to-frequency converter. The data is processed by industrial PC with National Instruments DAQ modules. The client-server distributed application written in LabView environment allows operators to control hardware and obtain measurement results over TCP/IP network.

THE PROBLEM STATEMENT

The project to construct a new accelerator complex NICA (Nuclotron-based Ion Collider fAcility) on the basis of the modernized existing Nuclotron accelerating facility is under active development at JINR. The modernization of the Nuclotron set the task to develop and put into operation a modern data acquisition and control systems. It is important to keep the existing functionality during accelerator runs while improving the graphic user interface, hardware control convenience, operation speed and stability. Fundamental principles of the control system organization are as follows:

- the client-server distributed model of data exchange,
- equipment operation reliability,
- applications operation stability in the long accelerator run.

The wide range of the acquisition electronics and software produced by National Instruments is quite satisfactory to solve the above tasks. A wide range of functions available in NI multifunction data acquisition modules allows to use only a few hardware modules to realize all necessary tasks. The software is based on .NET Framework, NI-DAQmx universal driver set and powerful graphics.

THE SUBSYSTEM STRUCTURE

The extracted beam diagnostics subsystem is intended to realize three features:

- to provide acquisition and remote observation of the extracted beam transversal profiles in four points of the transportation channel by means of proportional chambers;
- to measure the beam intensity using an ionization chamber;
- to control the gain of proportional chambers by means of their high voltage power supplies adjustment.

The equipment of the primary signal registration from the detectors is located on the beam transfer channel. The server computer and data acquisition modules are placed in the accelerator control room at about 400m distance from the detectors.

The subsystem structure is shown on Figure 1. The subsystem hardware has been made on the basis of Advantech industrial PC, multifunctional modules NI USB-6259 BNC, NI PCI-6703, NI SCB-68 produced by National Instruments [1], HV power supplies N1130-4 produced by WENZEL Elektronik and FEMTO variable gain sub femto ampere amplifier DDPCA-300 [3] with trans-impedance gain from 10^{+4} to 10^{+13} V/A. Additional custom voltage-to-frequency converter hardware to convert a voltage level signal from current amplifier to a frequency signal suitable for transfer of the data into the NI USB-6259 integration counter has been built.

HARDWARE OPERATION PRINCIPLES

Extracted beam intensity is calculated from measured ionization chamber current. The measured current is amplified by DDPCA-300 current amplifier and then converted to frequency signal by voltage-to-frequency converter. The signal is transferred over the long coaxial cable to 32-bit counter of USB-6259, synchronized by signal of extraction gate. The software allows to measure the cycle intensity and integral intensity of the extracted beam. The DDPCA-300 gain is controlled by 4 bit wide digital port of the USB-6259.

Beam transversal profiles are obtained by multiwire (32x32) proportional chambers. Signals from all 64 wires are multiplexed into a single analog line. Four ADC channels of USB-6259 are used to measure the analog signals from each proportional chamber. All ADCs are clocked by external clock signal formed by multiplexer thus allow deriving signals of each wire from the single line.

Proportional chambers gas amplification can be adjusted by means of high voltage power supplies which are controlled by NI PCI-6703 DACs. Adjusted high voltages are measured by four channels of USB-6259 ADCs.

SOFTWARE COMPLEX

The software complex consist of server application running on the server computer and few identical client applications which retrieve measurements data, visualize

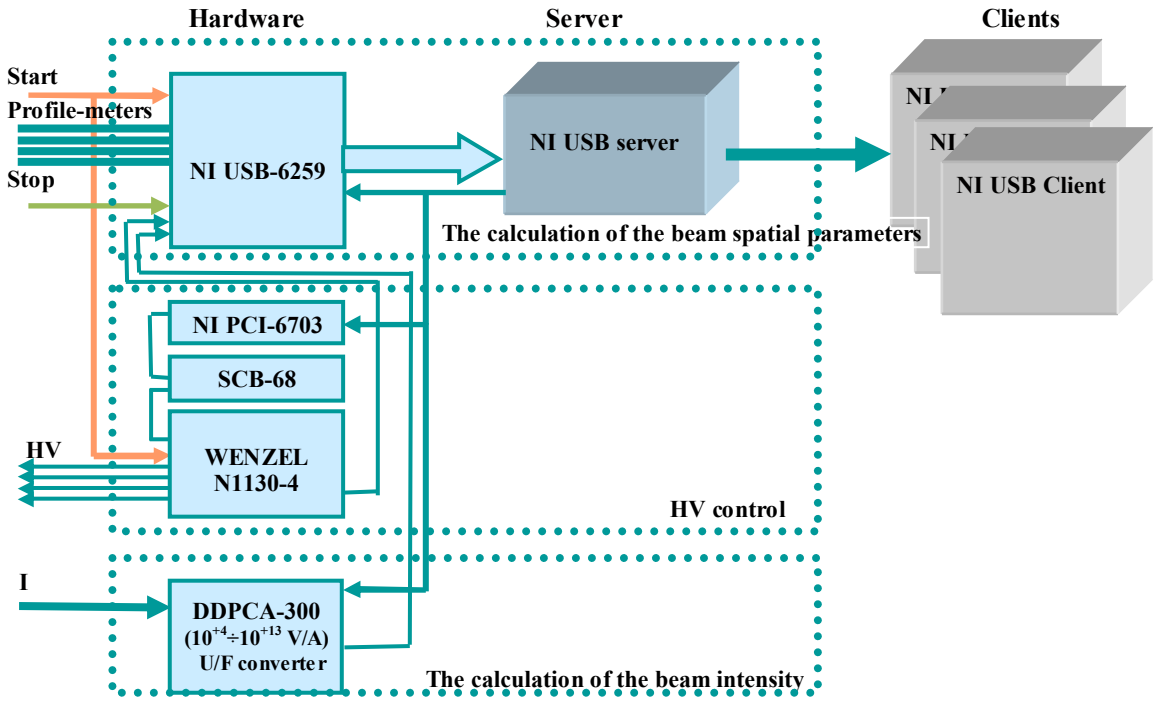


Figure 1: Extracted beam diagnostics subsystem structure.

it and allow an operator to control the hardware via TCP/IP network.

The server and clients applications are written in LabView development system for Windows. The data exchange between server and client applications are realized using LabView DataSocket server (DSS) running on the server computer. It realizes the application-layer DSTP transport protocol [2] operations and simplifies the task of organizing simultaneous interchange of data with many client applications connections in one cycle of the Server program. The publishing and subscribing of the data, located in the DSS URL-named memory, are preformed via a simple low-level interface of DataSocket Read and Write functions. These functions allow getting rid of the low-level TCP/IP programming and greatly simplify the server and client applications code. Besides, DataSocket server allows arranging data access permissions (write or read permissions) based on clients IP addresses. One of the client programs can be allowed to control the hardware, other are used only for the data visualization.

The server application consists of two continuously working cycles While and a Timed Loop cycle. In the first cycle, measurements of the beam spatial characteristics from proportional chambers and their power supplies voltages are acquired from NI USB-6259 module. The external ADC clock ensures correct timing of wires signals measurements. The second cycle reads-out the remote control DSS variables and makes a decision about hardware controls source - the server application or the controlling Client via DSS variables. The Timed Loop cycle is purposed to organize the start of the beam intensity measurements in a precisely given moment of time: 32-bit counter of the NI USB-6259 module counts the number of pulses of the voltage-frequency converter between the signals of the beginning and end of the slow extraction gate. At the moment when the counting is finished, the value on the counter is read-out, summed up with stored integral value and sent to DataSocket server together with time stamp and other measurements and calculations of the beam intensity. The integral intensity and the number of completed measurements cycles can be reset remotely.

The client application is intended for on-line and off-line processing of the measured data, its visualisation and remote hardware control. It consists of the two program loops: the Event structure and a continuous working cycle While.

The first cycle treats the events from all application controls such as buttons, sliders, drop lists and xcontrols. According to controls values, it sets the operation regime (online processing or visualisation of the offline data collected before), proportional chambers and current amplifier parameters, graphics scales and colors. The working cycle While connects to DataSocket server in online mode or open a file with previously stored data in offline mode and read the acquisition data. The obtained data is then processed to calculate instantaneous and integral beam profiles spatial statistical parameters such as beam position and dispersion taking into account the chosen distance between the wires in each profile-meter. 32 instantaneous beam profile shots for chosen point of the transfer line allowing to see the beam profile time progression and the integral beam profile are displayed in the «Profiles» tab for both transversal coordinates. The tab «Integral profiles» shows the integral profiles of all

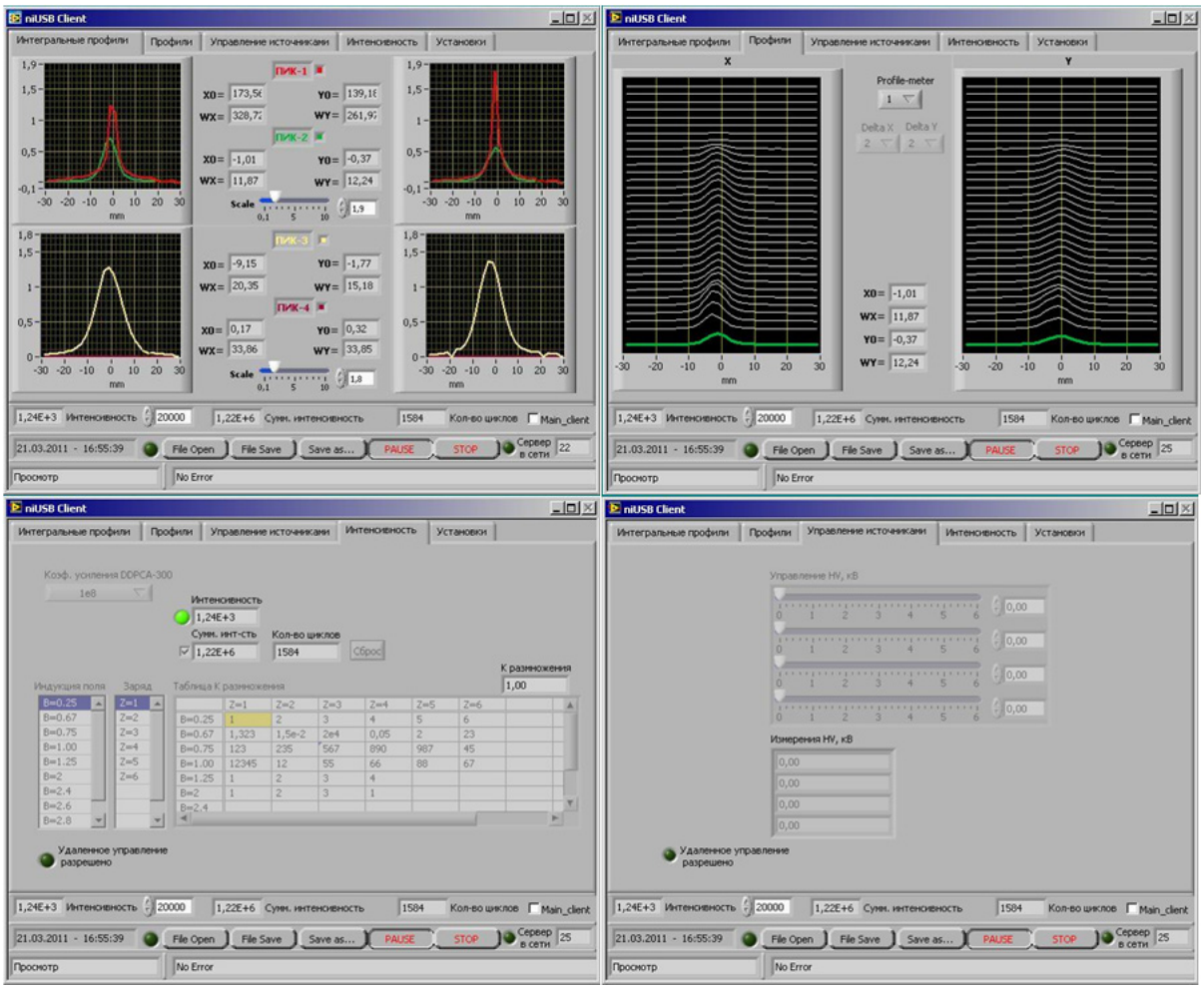


Figure 2: The beam tests of the subsystem to control extracted beam parameters (21.03.2011– 16:55). The Client window tabs: a) “Integral profiles”, b) “Profiles”, c) “Intensity”. d) “HV supply control”

profile-meters simultaneously. Intensity and high voltage measurements are visualised as well on corresponding application tabs.

The client software allows operator to control remote hardware such as current amplifier and proportional chambers high voltage power supplies if it is allowed by server. Otherwise the corresponding controls are disabled.

EXTRACTED BEAM DIAGNOSTIC SUBSYSTEM BEAM TESTS

The beam diagnostics subsystem was tested on the extracted beam during March 2011 Nuclotron run. Figure 2 shows window tabs of the client application. The “Integral profiles” tab illustrates integral transversal beam profiles while “Profiles” tab shows 32 momentary profiles shots from the selected multiwire detector in both X and Y coordinates. The “Intensity” tab shows extracted cycle intensity and integral intensity over 1584 cycles. Since the screen tab with the integral profile values must be watched permanently, the values of the intensities are given also below all the screen tabs. The last tab “HV supply control” contains HV power supplies controls and measurements. Hardware controls are disabled on all application window tabs as remote control was not allowed by the server.

CONCLUSIONS

The distributed subsystem to control parameters of the Nuclotron extracted beam was developed on client-server technology using National instruments acquisition modules, FEMTO current amplifier and LabView development system. Beam tests during March 2011 Nuclotron run showed the correct approach to the hardware-software selection and stable complex operation.

REFERENCES

[1] http://www.ni.com
[2] DataSocket Transfer Protocol (dstp) Overview. http:// http://zone.ni.com/devzone/cda/tut/p/id/3223
[3] http://www.femto.de/products/ddpca.htm

A CUSTOMIZABLE PLATFORM FOR HIGH-AVAILABILITY MONITORING, CONTROL AND DATA DISTRIBUTION AT CERN

M. Braeger, M. Brightwell, A. Lang and A. Suwalska, CERN, Geneva, Switzerland*

Abstract

In complex operational environments, monitoring and control systems are asked to satisfy ever more stringent requirements. In addition to reliability, the availability of the system has become crucial to accommodate for tight planning schedules and increased dependencies to other systems. In this context, adapting a monitoring system to changes in its environment and meeting requests for new functionalities are increasingly challenging. Combining maintainability and high-availability within a portable architecture is the focus of this work. To meet these increased requirements, we present a new modular system developed at CERN. Using the experience gained from previous implementations, the new platform uses a multi-server architecture to allow running patches and updates to the application without affecting its availability. The data acquisition can also be reconfigured without any downtime or potential data loss. The modular architecture builds on a core system that aims to be reusable for multiple monitoring scenarios, while keeping each instance as lightweight as possible. Both for cost and future maintenance concerns, open and customizable technologies have been preferred.

INTRODUCTION

The context of this work is the required replacement of an existing monitoring and control system at CERN: the Technical Infrastructure Monitoring system (TIM) takes part in the supervision of infrastructure surrounding the accelerator complex and is mainly used within the CERN Control Centre [1][2]. It has been in place for the last 5 years (TIM) and is now at an advanced stage of maturity. However, due to the use of now obsolete technologies, a new implementation was required, and this was an occasion to reassess the current and future use of the system, and update the design to meet these new challenges.

The services provided by TIM have evolved over the years. In addition to the direct monitoring of hardware, the software also provides real-time control of access systems and transmission of video signals. Such real-time services impose strict time constraints on the transmission of selected data. Moreover, due to the proven reliability of the service, an increasing number of external systems have come to rely on it: data re-distribution over various protocols is now one of the principal tasks of the system. The data is then being used in making critical decisions elsewhere.

With the evolutions described above, the availability of the system has become increasingly important, imposing strict constraints on any maintenance stops. This is accentuated by the variety of the data transiting through the system, since it cannot be linked to any particular phase of accelerator operation. One of the main aims during the redesign was to improve the availability of the old system – although already very high – and facilitate maintenance procedures.

The re-designed TIM system is based on the new CERN Control and Monitoring Platform (C^2MON*). The aim is to provide a generic platform that can be reused in multiple monitoring scenarios. A second service at CERN will also be based on C^2MON in the near future (the Diagnostic and Monitoring service or DIAMON [3]). More generally, a shared platform should help in harmonizing monitoring solutions used at CERN, optimizing costs, maintenance and system compatibility.

CONSTRAINTS ON MODERN MONITORING SYSTEMS

Modern monitoring systems are asked to meet ever more stringent requirements. Reliability has always been a key demand, as for all systems used on a 24/7 basis for operational monitoring: data should pass reliably through the system and be reliably correct! On the other hand, the handling of critical data can result in new availability constraints: external operational decisions may rely on this data. Moreover, these actions may even be triggered in an automatic way, with processes gathering data from the monitoring system through some external protocol. In addition to these availability constraints, performance in terms of data throughput and delivery speed must often be guaranteed. Any transmission of signals used for human interactions impose tight constraints on the time needed for data to pass through the system, including during data avalanches. A good example of this is data used for access control, when operators will expect a rapid response (typically under 1s). Finally, while satisfying these additional constraints, it is expected that the product will provide good maintainability qualities, in particular in terms of applying patches at short notice and the introduction of new features to satisfy evolving demands.

In view of the discussion above, the C^2MON platform aims to increase the availability and maintainability of the systems it is replacing, while maintaining the reliability of the older systems.

SYSTEM ARCHITECTURE: A GENERIC-PURPOSE MONITORING PLATFORM MAXIMISING AVAILABILITY

Having covered the background to this project and discussed current monitoring requirements in general, we now focus on the design of the C^2MON platform, and

* Further details from mark.brightwell@cern.ch

how the design choices have helped achieve the goals outlined above.

Generic: in what sense?

Many monitoring scenarios have similar requirements: acquiring data using a variety of protocols, managing the data (temporary storage, logging, distribution for instance), and displaying it for human visualization. Using a modular architecture, the C²MON platform was designed from the start to fit multiple monitoring scenarios: at CERN, it will be used as the basis for two distinct monitoring systems TIM and DIAMON.

The DAQ design allows modules to be written for new protocols and underlying hardware. The server architecture builds on a core part, to which optional modules can be added for custom behaviour. A client API is provided and can easily be reused in external GUIs for displaying C²MON data (an actual GUI implementation is also available). The only genuine constraint when using the C²MON platform is the data structures involved: for instance, internal data points correspond to single primitive values and need configuring as such in the database. Adding new data structures to the system remains possible, but requires a good understanding of the core design of the platform.

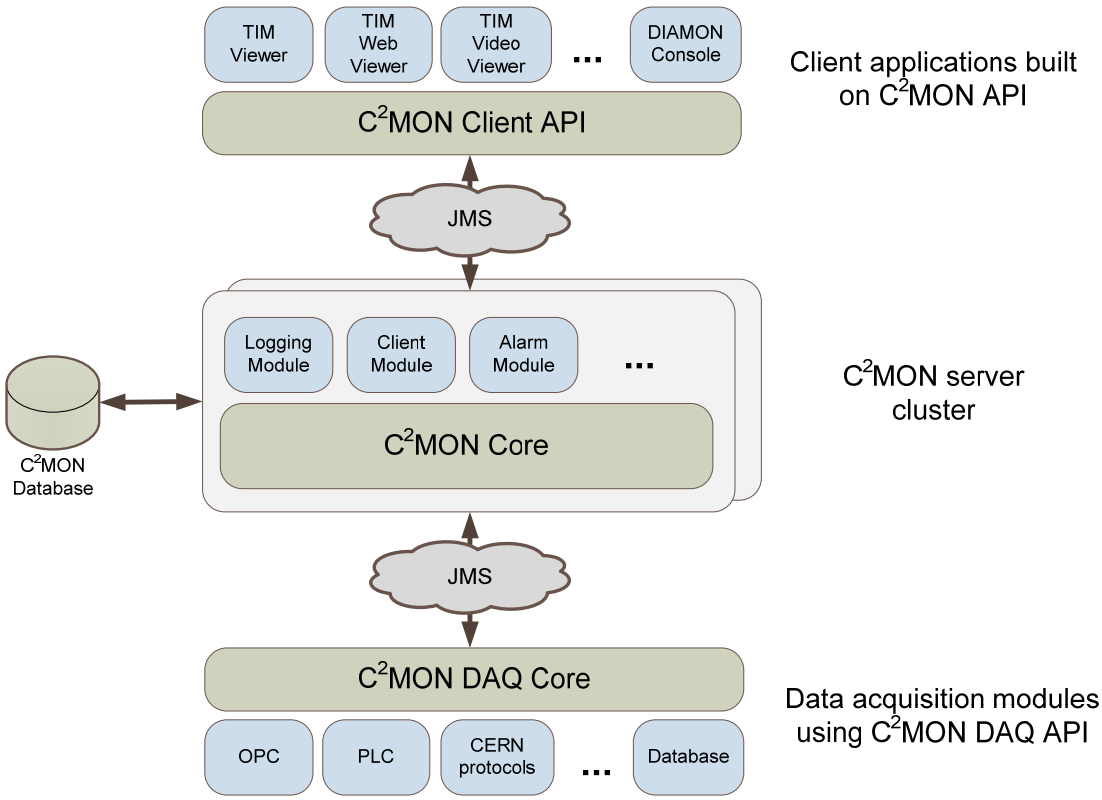


Figure 1: C²MON architecture overview

Architecture Overview

The C²MON platform follows a traditional 3-tier architecture, with data acquisition (DAQ), business logic (server) and client layers. An overview of the architecture is provided in figure 1 above. The DAQ layer is made up of any number of separate processes acquiring data through a variety of protocols. A DAQ implementation makes use of a common provided Core API for communicating with the business layer. The server layer itself consists in a cluster of servers. The server architecture is made up of a core part and a number of optional modules providing additional functionalities. The client layer provides a complete API to communicate effectively with the server. A number of client applications have been built on top of this API (e.g. the TIM Viewer). The various layers are linked with each other through messaging middleware.

The system is designed to run both in a single-server and multi-server configuration. In the multi-server configuration, data is shared between the servers in a distributed cache. Servers can be started on the fly and join an existing cluster. The multi-server configuration improves the availability of the overall system by adding failover support: in case of a single server failure, the remaining cluster is able to take over its workload. It also improves the maintainability of the system, since patches can often be rolled out across the servers without bringing down the whole cluster. The cache is regularly persisted

to the DB and can be recovered even after a complete cluster restart.

As already explained, running in a multi-server mode improves the availability of the C²MON platform. On the other hand, it comes at the cost of a more complicated design, and involves the use of more advanced/specialized software for the cache distribution (see the section below on technological choices). For this reason, the option of running in a single-server mode is of interest, particularly when occasional short downtimes are possible. This mode also remains as an emergency fallback scenario, rapidly deployable with minimum configuration settings.

Runtime System Data Configuration

As already mentioned, adding/removing or reconfiguring the monitored data is a frequent use-case for most monitoring systems. To maintain a high availability, the design must allow for these to be applied with minimal disruption to the service. To achieve this, the C²MON platform uses a flexible approach, allowing each data acquisition module to decide whether it is able to apply the changes with or without a restart of the particular acquisition process. Before providing some details, let us briefly review the C2MON re-configuration process.

All the configuration details for the monitored data points are held in an *offline* database. Before submitting this data to the online C²MON system, it must pass a strict data validation process. This process has proven as important as the system design to the long-term success of the service, but will not be discussed further in this paper. Once the data is correctly configured offline, it can be passed to the C²MON system for online configuration. On reception of a configuration request, the C²MON server loads the details from the offline database. The changes are applied internally, persisted in the online DB and made available in the distributed cache. These changes are applied immediately across all servers in the cluster. Moreover, any changes affecting the data acquisition layer will be propagated down to the appropriate DAQ modules. The functionality available at this level will then depend on the particular DAQ module implementation (specific to the protocol or hardware): if any changes cannot be applied at runtime by the module, the server will be informed that a DAQ process restart is necessary for the changes to take effect.

Using Real-time Filtering on the DAQ Layer

Guaranteeing the availability of a monitoring system also involves protecting it from overload during data avalanches. The C²MON system already greatly restricts the amount of data sent through the system by filtering out redundant values at the DAQ layer (these are forwarded to a separate statistics-gathering process for system administration purposes). However, to help protect the system further, the DAQ layer provides a *dynamic-filtering* option, whereby time dead bands are imposed on individual points detected as feeding too much data to the C²MON server cluster (various strategies on measuring the data throughput are provided).

Technological Choices

The platform is implemented in Java. Based on previous experience, the technological choices were guided by the following principals:

- Stick to proven technologies, facilitating project maintenance and stability.
- Where possible, use open-source resources, facilitating the reuse of the software in other projects. If justified by performance, use other 3rd party products, but provide a basic open-source fallback option.

With these principals in mind, the following choices were made:

- Spring is used as the basic framework wiring together the various components.
- Concerning the JMS middleware, implementations of the various C²MON layers are available for ActiveMQ only.
- The system was designed against an Oracle database, but should run with other databases (this has not been tested so far; some transaction support is required).
- Database persistence is done using the iBatis Java library.
- The cache library used is Ehcache, an open-source Java cache.
- To run in a multi-server configuration, the cache is distributed using the Terracotta technology, which involves running a separate Terracotta server (same company as Ehcache). This technology remains free when run in a basic configuration (sufficient for most purposes).

CONCLUSION

The C²MON platform has been designed over the last year with a view of reusing it in multiple monitoring scenarios. With this in mind, care was taken to provide a flexible core, with convenient hooks for adding custom functionalities. Together with the essential reliability of such a system, the design has attempted to maximize its availability, without affecting maintainability. We list the important design choices made in achieving this:

- The option of running a cluster of servers using a distributed cache, providing a failover mechanism and allowing rolling application of software patches.
- Runtime configuration of the monitored points/ hardware (so without server/acquisition process restarts).
- No operational database dependencies (only required for reconfiguration functionality).

All core parts of the system rely solely on open-source & free technologies.

The C^{2}MON platform is currently in the testing phase, with a planned rollout into production in December 2011.

REFERENCES

[1] U. Epting, P. Ninin, R. Martini, P. Sollander, R. B. B. Vercoutter and C. Morodo, "CERN LHC technical infrastructure monitoring", ICALEPCS'99, Trieste, 1999, https://edms.cern.ch/document/394483/1.

[2] J. Sowisek, A. Suwalska and T. Riesco, "Technical infrastructure monitoring at CERN", EPAC'06, Luzern, 2006, https://edms.cern.ch/document/750284.

[3] M. Buttner, P. Charrue, J. Lauener and M. Sobczak, "Diagnostic and Monitoring CERN Accelerator Controls Infrastructure - The DIAMON Project - First Deployment in Operation", ICALEPCS 2009, Kobe, 2009.

SOFTWARE AND CAPABILITIES OF THE BEAM POSITION MEASUREMENT SYSTEM FOR NOVOSIBIRSK FREE ELECTRON LASER

E.N. Dementyev, A.S. Medvedko, S.S. Serednyakov, E.N. Shubin, V.G. Tcheskidov N.A.Vinokurov.
BINP SB RAS, Novosibirsk, Russia

Abstract

The system that measures the electron beam position in Novosibirsk free electron laser with the application of electrostatic pick-up electrodes is described. The measurement hardware and main principles of the measurement are considered. The capabilities and different operation modes of this system are described. In particular, the option of simultaneous detection of accelerated and decelerated electron beams at one pick-up station is considered. Besides, the operational features of this system at different modes of FEL performance (the 1st, 2nd, and 3rd stages) are mentioned.

INTRODUCTION

A high-power free electron laser (FEL) based on the accelerator-recuperator [1] is under construction now at Budker Institute of Nuclear Physics. The first and second phases of the project were commissioned recently. The beam position measurement system is one of the important diagnostic systems of the FEL. The system consists of a large amount of electrostatic pick-up electrodes installed on the vacuum chamber of the microtron-recuperator and special measuring devices located outside the accelerator hall. The main characteristics, measurement algorithm and capabilities of this system are described in this paper.

SYSTEM STRUCTURE

As mentioned above, the main part of the system is a number of pick-up electrodes (BPM stations) installed in different parts of the vacuum chamber of the microtron-recuperator. Each BPM station has four buttons with a clear aperture diameter of about 80 mm. Signals from the plates of all BPMs are transmitted by cables outside the shielded hall to the site of location of the measurement electronics. The beam position is determined via simultaneous measurement of the amplitudes of signals induced on the buttons by the beam field. It is done with the use of the following linear approximation formula:

$$x = k \frac{U_+ - U_-}{U_+ + U_-},$$

where k is a factor with the dimension of length. It is obtained from electromagnetic measurements (in the case of a circular tube, $k \cong a/4$, where a is the tube radius); U_+ and U_- are the amplitudes of signals from a pair of opposite (left and right) plates.

Due to the principle of operation of the microtron-recuperator, there are simultaneously several electron beams at some places of beam propagation. For example, accelerated and decelerated beams are moving simultaneously in the accelerating RF section. The total number of different beams in the RF section depends on the number of beam turns and equals *2*N* (where *N* is the number of turns). Owing to this feature, the task of separated measurement of the positions of different beams at the same BPM station arises. This task influenced the choice of the measurement hardware and the measurement algorithm.

The main devices used in this system are as follows:

- a 4-channel analog-to-digital converter with a 32k-memory cell per each channel and external start,
- a 4-channel delay line with a step of time delay adjustment of 250 psec, and
- a 5-channel multiplexer

All these devices are manufactured in the CAMAC standard. A more detailed description can be found in [2].

The principle of measurement is shown in Fig. 1.

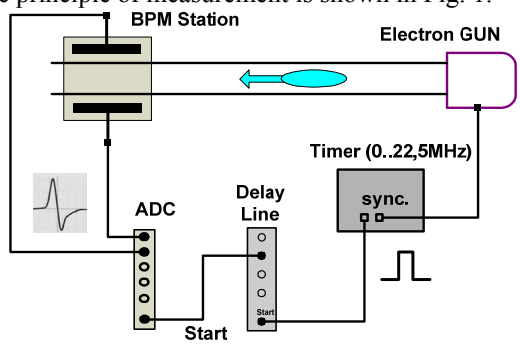


Figure 1: Scheme of beam position measurement.

The measurement is performed bythe 4-channel ADC, to which 4 cables from the BPM plates are connected. The moment of measurement is determined by an external "Start" signal. This signal is generated by the Timer – a device that also sends a control signal to the electron gun modulator. Thus the frequency of the "Start" signal is equal to the frequency of the movement of the electron beams. For compensation of the difference in the time of movement of all these signals and time of beam movement in the vacuum chamber, the "Start" signal is passed through the adjustable delay line. A precise adjustment of the time delay value allows one to select the time of measurement, tune the measurement moment to the pulse maximum, and also to tune the measurement to different beams, moving one after another. Since the number of BPM stations is rather large, the measurement devices are grouped in units. Each unit contains one ADC and five multiplexers, so it is possible to connect five BPMs. Besides, each unit uses one channel of the delay line. So, since a CAMAC crate can hold up to three units, there should be one delay line in each CAMAC crate. The entire scheme of this system, connections of the BPM electrodes and measurement devices are presented in Fig.2.

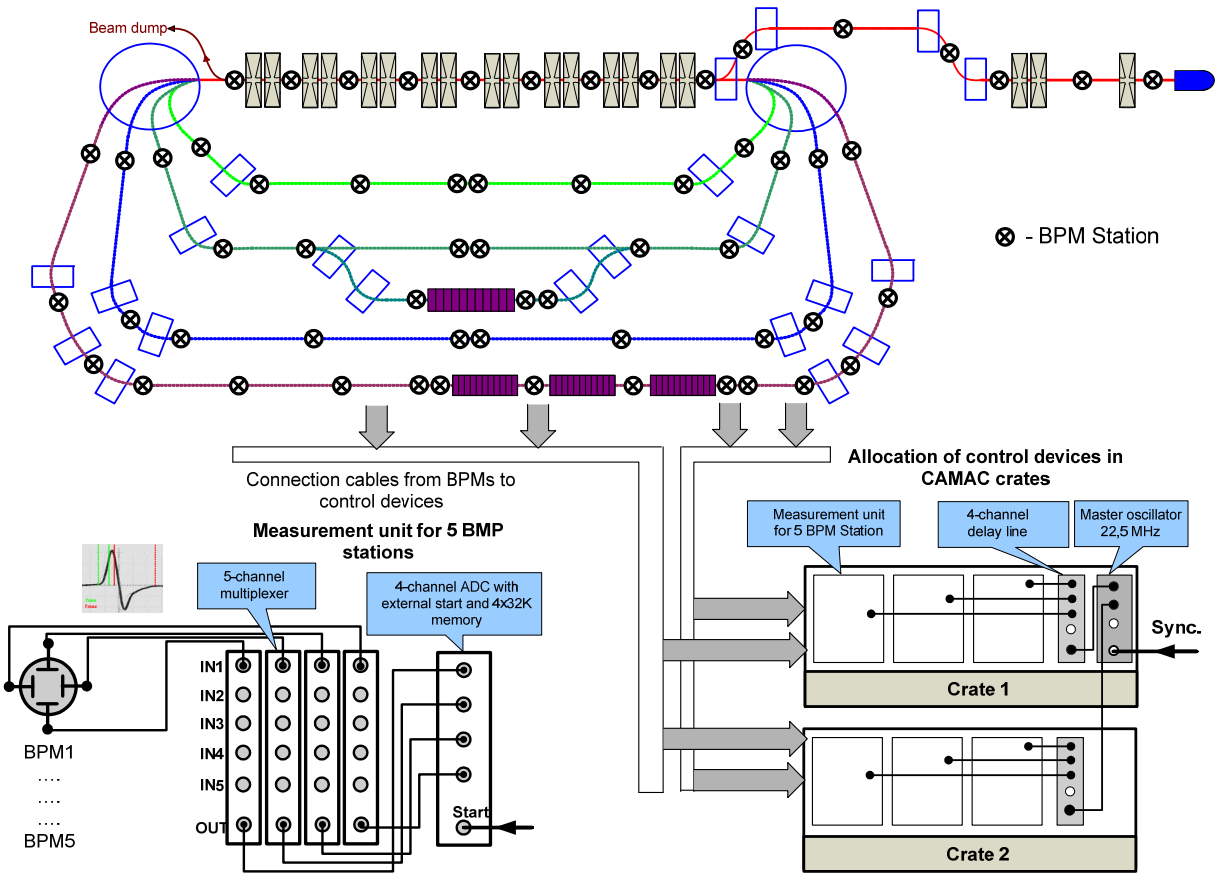


Figure 2: Main scheme of the beam position measurement system.

SOFTWARE

The control software for this system is a single application running on the IBM-PC computer. Communication with the CAMAC crates is realized with the help of the ISP controller, developed at Budker INP. The application can operate in several different regimes.

The first regime is a serial poll of all the BPM stations for the purpose of determination of the transverse coordinates of the beam (the main operating regime). In this regime, the application is subsequently polling all BPM stations. Measurement at each BPM station is executed in a specified delay time range assigned to the delay line. After running throughout the range, the program finds the maximum of signals from each button, computes the transverse coordinates, and displays it either as the beam trajectory or as the beam positions at each station (see Fig.3). In this case, measurement of one point requires ~15 CAMAC NAF instructions and a time of approximately 1 millisecond. Usually, each BPM station has a range of about 10-15 time delay points. So, for the FEL 1st stage with 37 measurement points in total, the overall time of measurement of the entire beam trajectory is 400-600 msec.

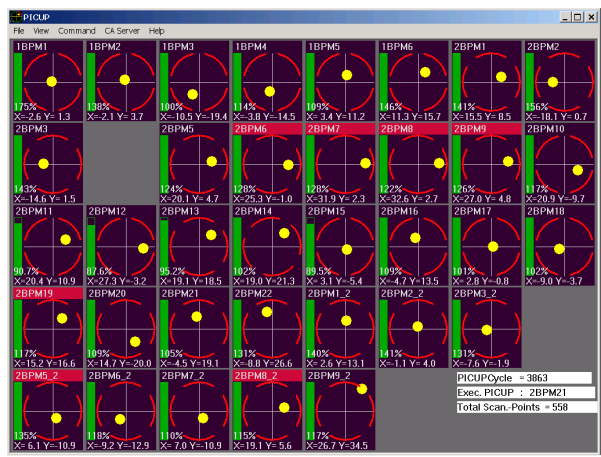


Figure 3: Window of a serial poll of the BPM stations.

Besides the calculation of transverse coordinates, in this regime the application calculates the sum of all maximum measured signals since this value is proportional to the value of electron beam current. The user can select any BPM station, and the application will calculate the beam current value in percents relative to this value on the selected BPM. The "relative" beam current value is drawn in the main window with green vertical bars. With this feature it is very easy to find areas of the accelerator track with beam current losses.

The second operation regime is a scanning throughout the delay time range of one of the BPM stations. While

executing this cycle, the program plots the time dependence graph of the beam-generated pulse. If there are several different beams at a selected BPM station, the user can view them in the same graph. The resulting plots of this regime are shown in Fig. 4. This regime is mainly used for specification of the time delay limits for scanning in the main, first regime. The user can manually select the time delay range as narrow as possible to reduce the time of processing of this BPM station. At the same time, this range should include all the four maximums of signals from the BPM buttons.

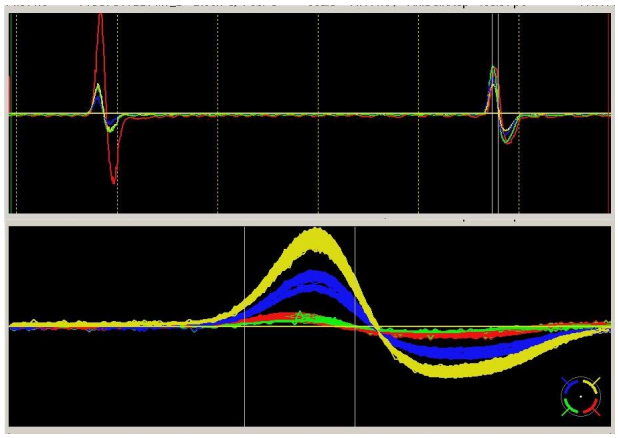


Figure 4: Graphs of the scanning cycle throughout the time delay range.

Another application of this regime is the study of the time stability of electron beam. The user can specify the scanning range as small as possible, containing only pulses from the buttons (see the lower part of Fig.4.). It is also possible to run this scanning cycle infinitely, without erasing the previous graphs. In this case, if there are some oscillations of beam position or beam current, the BPM button pulse graphs will cover the whole area of these oscillations.

The third and very useful regime is a poll of the waveform of the BPM button pulses with the frequency of the beam movement and a constant time delay value. As in the previous regimes, measurement is triggered by an external signal. However, in this case, measurement data are stored in the internal ADC. The ADC has a memory of 32 K of 12-digit words for each channel and can perform measurements with a frequency as high as 50 MHz. When the whole memory is filled, the application reads all the measurement data from it and plots them (see Fig. 5). Measurements in this regime being executed with the beam movement frequency, the amplitude, frequency, and shape of beam pulse oscillations, if any, can be found. Besides, modifying the beam pulse area where these measurements will be performed, one can easily study the nature of oscillations of beam positions or current.

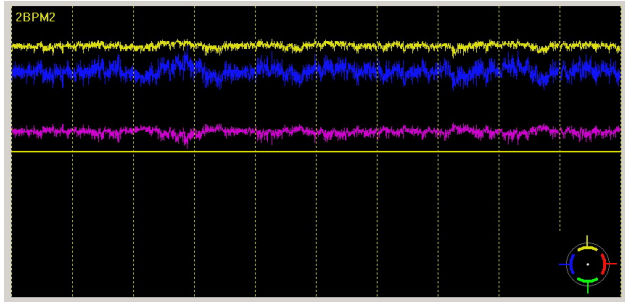


Figure 5: Plots of measurements with a constant time delay value and beam movement frequency.

Another useful feature of this application is the ability to transfer all measured data (the x and y coordinates and beam current) to another application in the FEL control LAN. For this purpose, a portable Epics Channel Access server is built-in inside the application. Every measured value is presented by one process variable.

CONCLUSION

The above-described system has been operating for about 8 years, demonstrating high reliability. The features of the measurement equipment allow observing simultaneously the accelerated and decelerated beams in the accelerator-recuperator beam-line. The interface and capabilities of the control software allow analyzing the behavior and time characteristics of the electron beam. The existence of the Epics Channel Access server in the program allows archiving all measured data and using them in other applications.

REFERENCES

[1] N.A. Vinokurov et al. Novosibirsk free electron laser facility: Two-orbit ERL with two FELs. Proceedings of FEL2009, TUOD01

[2] E.N.Dementiev, N.I.Zinevich, E.I.Shubin, “Atomic Energy”, v.93, iss.6, December, 2002, pp.453-455.

PROGRESS OF THE TPS CONTROL SYSTEM DEVELOPMENT

Jenny Chen, C. H. Kuo, C. Y. Wu, Y. S. Cheng, P. C. Chiu, C. Y. Liao, K. H. Hu, Y. T. Chang, Demi Lee, S. Y. Hsu, C. S. Wang, Y. K. Chen, Y. R. Pan, K. T. Hsu
NSRRC, Hsinchu 30076, Taiwan

Abstract

The Taiwan Photon Source (TPS) is a low-emittance 3-GeV synchrotron light source which being in construction at National Synchrotron Radiation Research Center (NSRRC) campus. Control system for the TPS is based on EPICS framework. Standard hardware and software components were defined. Prototypes of various subsystems are in implementation. Event based timing system was adopted. Power supply control interface accompany with orbit feedback support are also defined. Machine protection system is in design phase. Integration with the linear accelerator system which are installed and commissioned at the temporary site for acceptance test was already done. Interface to various system are still in negotiate stage. Development of the infrastructure of high level and low level software are on going. Progress will be summarized in the report.

INTRODUCTION

The TPS [1] is a latest generation of high brightness synchrotron light source which is being in construction at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan. It consists of a 150 MeV electron linac, a booster synchrotron, a 3 GeV storage ring, and experimental beam lines. Civil construction has started from February 2010. The construction works are scheduled to finish in later 2102. Accelerator system installation and system integration will be proceeding in the year of 2013. Commissioning with beam is scheduled in the first quarter of 2014. Control environment should be ready in 2013 to support subsystem integration test and hardware commissioning without beam. Control system for the TPS is based on the EPICS framework [2]. The EPICS toolkit provides standard tools for display creation, archiving, alarm handling and etc. If users have found these tools inadequate, development of in-house alternatives is feasible and compatible. The big success of EPICS is based on the definition of a standard IOC structure together with an extensive library of driver software for a wide range of I/O cards. Many users of the system report a steep learning curve and the need for significant development resources, but this is balanced by the large installation base and proven ability of this approach. The EPICS toolkits which have various functionalities will be employed to monitor and to control accelerator system. The control system consists of more than a hundred of EPICS IOCs. The CompactPCI (cPCI) IOC will be equipped with input/output modules to control subsystems as standard IOC or the TPS control system. The power supply and fan module of the cPCI crate will be hot-swapped. Adopting cPCI platform for EPICS IOCs provides us a chance to take advantages of local IT industry products with better supports and low cost. The other kinds of IOCs are also supported by the TPS control system, such as BPM IOC, PLC IOC, various soft-IOC and etc. Consoles and servers are PCs or blades PC running Linux. To achieve high availability of the control system, emphasis has been put on software engineering and relational database for system configurations. Data channels in the order of 10^5 will be serviced by the control system. Accessibility of all machine parameters through control system in a consistent and easy manner contributes to the fast and successful commissioning of the machine. High reliability and availability of TPS control system with reasonable cost and performance are expected.

CURRENT STATUS

Major procurement is scheduled in 2011 to 2012. The design for the control system environment is already frozen in the hardware part. The software environment will be completed in 2012. Control related applications are started and expected to a preliminary form before subsystem ready in early 2013. Define TPS control system standard processed during the last couple of years. Efforts of the control system development status are summarized in following paragraphs. Progress on various issues is summarized in following paragraph.

Networking

Mixed of 1/10 Gbps switched Ethernet will be deployed for the TPS control system [3]. The Gigbit Ethernet connection will be delivered at edge switches installed at control and instruments area (CIA). One CIA corresponding to one cell of the storage ring, there are 24 CIAs total. The control network backbone will be 10 Gigabit link to the control system computer room and redundancy network installation site in the specific CIA #24. Private Ethernet is used for Ethernet based devices access which will support fast Ethernet and GbE. Adequate isolation and routing topology will balance between network security and needed flexibility. The file and database servers are connected to the control and intranet network, allowing the exchange of data among them. Availability, reliability and cyber security, and network management are focus in the design phase.

Equipment Interface Layer

There are several different kinds of IOC at equipment interface layer to satisfy various functionality requirements, convenient, and cost consideration. Most of the devices and equipments will connect to cPCI IOCs crates running EPICS directly. The 6U cPCI platform was chosen for the EPICS IOC platform. Local company manufactured crate and CPU module providing an economic solution is the major reason. To simplify various developments at construction phase, only 6U modules will be supported for the machine control system. The cPCI EPICS IOC equipped with the latest generation CPU board will be standardized as ADLINK cPCI-6510 CPU module [4]. The CPU module equipped with Intel Core i7 CPU running Linux delivery high performance to meet various applications. The latest releases version of Fedora core Linux distribution is adopted at IOC level. The cPCI-7452 128 bits DI/DO module will be used for BI, BO solution, this high density version in 6U form factor will save crate slots and satisfy most of applications. Industry pack (IP) carrier board in 6U cPCI form factor can equip up to 4 IP modules. Various IP modules are adopted for necessary applications. ADC and DAC modules in IP module form factor will be used for smaller channel count application, such as insertion devices control. Event system modules are in 6U cPCI form factor. Private Ethernet will be heavily used as field-bus to connect many devices. Power supplies of all magnets except for correctors are equipped with Ethernet to the EPICS IOC communication. Multi-axis motion controller with Ethernet interface will be the standard support for the control system.

Ethernet attached devices will connect to the EPICS IOC via private Ethernet. Some network attached devices might cannot work properly when the network traffic busy due to its simply TCP/IP stack implementation. Private network ensure lower traffic to ensure the reliability and performance of the links. Devices support VXI-11, LXI, Raw ASCII, Modbus/TCP protocol can connect to the EPICS IOC via TCP/IP interface directly. Devices of this category include power supply, temperature acquisition (RTD or thermocouple), digital multi-meters, oscilloscopes, signal generator, and other instruments.

High resolution ADC module embedded with EPICS IOC from D-tAcq [5] will be used for the analogue signal reading, it provides 10 Hz rate reading and has transient capture buffer up to 100 Ksample/sec sampling rate.

All corrector power supply will interface by the corrector power supply controller (CPSC) module. The CPSC equip with 20 bits DAC and 24 bits ADC. Corrector power supplies will be driven by the output of this module. Two SFP ports supported by the on board FPGA (Spatan 6), these SFP ports will receive correction setting (Autora and Gigabit Ethernet by using UDP/IP protocol) from fast orbit feedback FPGAs ot slow orbit feedback PC, feed-forward correction computer and IOC. Setting command received by these SFP ports will be added with the slow setting from EPICS CA setting.

Power Supply System Control

TPS power supplies control interface are divided into three categories rather than a unified solution [6]. All of the power supplies will be provided by three different vendors. The reason of this choice is to meet the practical situation from budget and available vendors and manpower.

The small power supply for corrector magnets, skew quadrupoles in the range of ±10 Amp categories. This category power supply will be in module form factor. Each power supply subrack can accommodate up to 8 power supply modules. A custom design CPSC module will be installed at control slot of the sub-rack. The CPSC will be embedded with EPICS IOC and provide fast setting SFP ports to support orbit feedback functionality. Power supply modules installed at the same sub-rack will interface to this CPSC module. The CPSC installed 20 bit DAC and 24 bit ADC to ensure necessary performance for the orbit control especially in vertical plane of the storage ring. To simplify the type of power supply, this category power supply will be used for the corrector of LTB/BTS/Booster Synchrotron, and storage ring as well as skew quadrupole magnet power supply. The CPSC modules support waveform capability which can support corrector ramping for the booster synchrotron if necessary.

The intermediate power supply with current rating 250 Amp will be equipped with Ethernet interface. Power-supplies are expected to have internal data buffer with post-mortem capability. Output current of the power supply will output at rear plane BNC connector, which can connect to the cPCI ADC module also. There are two versions of power supply in this category, sextupole power supply with 16 bits resolution and quadrupole power supply with 18 bits resolution DAC. Both kinds of power supply can meet the 50 ppm and 10 ppm performance specifications for long-term drift. Both kinds of power supply noise level are in the 10 ppm range of the full scale

The storage ring dipole DC power supply and power supplies for the dipole and quadrupole power supply of the booster synchrotron had already contracted to the IE Power (Acquired by Eaton in 2011). The control interface will be a serial interface. It is expected that a serial and Ethernet adapter enable directly Ethernet connection. Control resolution of these power supplies will be 18 effective number of bits, noise and drift will be better than 10 ppm of these power supplies. The dipole and quadrupole power supply of the booster synchrotron have built in waveform support with external trigger capability. This functionality is essential for energy ramping of the booster synchrotron. All of these power supplies will interface with the EPICS IOCs directly.

Timing System

The event system consists of event generator (EVG), event receivers (EVRs) and a timing distribution fiber network [7, 8]. EVG and EVRs can be installed with various universal I/O mezzanine modules to meet different input/output requirements. The mechanical form factor of EVG and EVRs is in 6U cPCI module. The 125 MHz event rate will deliver 8 nsec coarse timing resolution. Fine delay is also supported by UNIV Output module with fine delay and special EVR which can generate gun trigger signal. Its high resolution and low timing jitter allow accurate synchronization of hardware and software across the TPS control system. Its usage simplifies the operation of the machine and allows complex sequences of events to be carried out by changing very few parameters. Prototype system was delivered in March 2011 already and applied to the TPS linac system commissioning at the temporary site successful. Consideration for the injection sequence control is also in intensive study.

Insertion Devices Control Interface

Insertion devices (ID) control for the phase I project include one set of EPU46, two sets of EPU48 and seven sets of in-vacuum insertion devices (two sets of 2 meter long IU22, three set of 3 meter long IU22, and one set of 3 meter long IU 22 with taper functionality). The driven mechanism of EPU46 and EPU48 are drive by servo motors. All IU22 drive by stepping motors. The motion controller is Galil DMC-404x Etherent/Series Accelera motion controller. Preliminary version of EPICS devices supports for this motion controller is in use. Improve is under way. The cPCI EPICS IOC equips with AI/AO/BI/BO I/O modules will serve each ID. All IDs will equip with SSI optical encoder. All encoder are connected to the motion controller. All parameters of motion controller will map to EPICS PV. Update rate up to 200 Hz is feasible. This might useful for feed-forward compensation which need to know the encoder value as fast as possible.

Diagnostic System Interface

New generation digital BPM electronics [9] is equipped with Ethernet interface for configuration and served as EPICS CA server with 10 Hz data rate. Another multi-gigabit interface will deliver beam position for fast orbit feedback purpose at rate up to 10 kHz. The BPM electronics will also provide post-mortem buffer for orbit analysis during specific event happened like beam loss. Post-mortem analysis can help to find the weakest point and provide information to improve system reliability.

High precision beam current reading and lifetime calculation will be done at a dedicated IOC. This IOC will install EVR to received booster cycle timing signals and high resolution IP ADC modules to digitize the DCCT signal and perform beam lifetime calculation.

The GigE Vision digital cameras will capture images for diagnostic purposes and other applications.

Counting type and integrating type beam loss monitors will be connected to the control system by counter or ADC modules installed at IOCs.

Feedbacks and Feed-forward Plans

Since the TPS adopt aluminium chamber with 4 mm in thickness. Eddy current effects prevent the standard corrector from fast corrector. Four fast correctors will be installed at bellows site for each cell. So, the global orbit feedback system by using slow and fast correctors in the same feedback loop is the baseline design [9]. A counter rotate multi-gigabit links will circulate 10 kHz rate orbit data among all BPM platforms. The FPGA module embedded in the BPM platforms will be configured as distributed fast orbit feedback engines. All of these platforms will be installed with EPICS interface for various feedback supports such as PID parameters, and matrix download. The correction command from FPGAs will be sent to the CPSC module located at the power supply sub-rack via Aurora links. The data buffer in the FPGA module can be used to capture fast orbit data up to 100 sec at 10 kHz rate for feedback system diagnostic and performance study. The capture can be triggered either software or hardware with post-mortem capability.

The slow orbit feedback is planned to use one dedicated EPICS IOC for each plane (maybe one aTCA compute blade for each plane). The orbit data can be acquired by the EPICS CA access or from the BPM grouping data directly by UDP/IP link. The slow orbit feedback will used the slow transfer matrix to calculate the correction values. The slow feedback loop will read the DC value of the fast corrector regularly (1 ~ 10 times/sec), multiply the inverse of the fast response matrix to get the orbit distortion which cause due to the DC setting of fast correctors. The same orbit distortion will transfer to the setting value by the slow correctors. This scheme can ensure the setting of the fast corrector always around zero. This will keep the maximum dynamic range of the fast corrector to suppress fast perturbation. The corrector setting for the slow orbit feedback loop is via UDP/IP packet. All setting value of the correctors are packed in one UDP/IP package. Each CPSC module extract their setting form specific offset within the packet. One packet can serve for all slow corrector in one plane for the whole ring.

The bunch-by-bunch feedback system will adopt signal processor based on the latest generation FPGA processor with embedded EPICS IOC and build-in diagnostics.

Turnkey System Solution

Turnkey systems such as linear accelerator and RF transmitters were delivered by industry as turnkey contracts and EPICS control environment. Work on EPICS support for the turnkey system with proprietary control environment is also not a problem. It is expected

that the turnkey systems delivered no matter with or without EPICS control system can integrate with the TPS control system. The linac system delivered in June 2011 and done acceptance at accelerator test site. It will move to the TPS linac site in later 2012 after civil construction of the TPS building is completed. Two RF transmitters were received in 2010. EPICS ready interface of these delivered system proved that the integration is not a big problem.

PLC and Interlock Solution

Current TPS control system will support Siemens S7-300 or compatible model from VIPA PLC which are delivered from the turn-key vendors. Yokogawa FM3R PLC will be used for most of control system related interlock system. FM3R with embedded EPICS IOC will be used for some applications also. Each subsystem is responsible to build their own interlock and protection system based on their requirement and preference. The global machine interlock system will collect various interlock signals from local interlock subsystem of orbit, vacuums, front-ends, radiation dosage monitors and etc. The beam disable commands to trip beam or inhibit injection can be distributed to the specific devices or subsystem by the global machine interlock system or uplink functionality of the event system.

Operator Interface

The operator interface level consists of Linux PCs for consoles and servers for various purposes. Various EPICS OPI tools and Matlab will be adopted for OPI development [10, 11]. It is planned currently that most of the GUIs will be implemented by EDM and Matlab. Consoles in the control room have multiple LCD screens. The OPI computer will be installed at the equipment area of control room with optical PCIe extension to remote display unit at control room. This can provide better cooling for the computer, reduce loudness at control room and provide clean control consoles. Large screen format displays hang on the roof at control room will be available for display of important parameters likes beam current, lifetime, vacuum distribution, synchrotron radiation image and etc.

Control Applications

Generic applications provided by the EPICS toolkit will be used for all kinds of applications. Standard tools such as the archiver, alarm handler and save/restore tools are supported. Channel Access (CA) is used as an interface for machine process variables (PVs) access. Simple tasks such as monitoring, alarm handling, display and setting of PVs are performed using EDM panels and strip tools. Cold start, warm up and shutdown process will done by Matlab scripts.

Physics Applications Interface

The accelerator physics tools for TPS include extensively adopted Matlab Middle Layer (MML) and Accelerator Toolbox (AT) software packages. It enables various developed applications of different machines to be directly adopted for TPS applications. The MML has also provided a systematic way of managing machine data, and is easily extended to separate data from transfer lines, booster and storage ring. The Middle Layer communicates with the machine from within MATLAB using LabCA. To enable early testing of the physics tools through the control system, a virtual accelerator has been implemented to give simulation of the accelerators through the EPICS PV interface. It was developed by provided EPICS device support to interface the model using the AT and MML.

SUMMARY

Implementation of the TPS control system is in on going. Major procurement is in proceeding. Assembly and system integration is scheduled in 2012-2013. The TPS control system will take advantages of the latest hardware and software technology to deliver better performance and functionality, avoid quickly obsolesce and more economic. There are many issues required further efforts and focus including relational data base, system configuration and operational management, various application programs. Identify priority and find an adequate measure to deal with various issues is current efforts.

REFERENCES

[1] TPS Design Book, v16, September 30, 2009.
[2] EPICS: http://www.aps.anl.gov/epics/index.php.
[3] Y. T. Chang, et al., “Infrastructure of Taiwan Photon Source Control Network”, These Proceedings.
[4] ADLINK website: http://www.adlink.com.
[5] D-Tacq website: http://www.d-tacq.com.
[6] C.Y. Wu, “Power Supply Control Interface for the Taiwan Photon Source”, These Proceedings.
[7] Micro Research Finland website: http://www.mrf.fi.
[8] C.Y. Wu, et al., “Timing System of the Taiwan Photon Source”, These Proceedings.
[9] C. H. Kuo, “BPM System and Orbit Feedback System Design for the Taiwan Photon Source”, These Proceedings.
[10] Y.S. Cheng, “Preliminary Design and Integration of EPICS Operation Interface for the Taiwan Photon Source”, These Proceedings.
[11] C. Y. Liao, “Image Acquisition and Analysis for Beam Diagnostics Applications of the Taiwan Photon Source”, These Proceedings.

OVERVIEW OF THE SPIRAL2 CONTROL SYSTEM PROGRESS

E. Lécorché, P. Gillette, C. Haquin, E. Lemaître, G. Normand, C.H. Patard, L. Philippe, D. Touchard, Ganil / Caen, France
J.F. Denis, F. Gougnaud, J.F. Gournay, Y. Lussignol, CEA-IRFU / Saclay, France
P. Graehling, J. Hosselet, C. Maazouzi, CNRS-IPHC / Strasbourg, France

Abstract

Spiral2 whose construction physically started by the beginning of this year at Ganil (Caen, France) will be a new Radioactive Ion Beams facility to extend scientific knowledge in nuclear physics, astrophysics and interdisciplinary researches. The project consists of a high intensity multi-ion accelerator driver delivering beams to a high power production system to generate the Radioactive Ion Beams being then post-accelerated and used within the existing Ganil complex. Resulting from the collaboration between several laboratories, Epics has been adopted as the standard framework for the control command system. At the lower level, pieces of equipment are handled through VME/VxWorks chassis or directly interfaced using the Modbus/TCP protocol; also, Siemens programmable logic controllers are tightly coupled to the control system, being in charge of specific devices or hardware safety systems. The graphical user interface layer integrates both some standard Epics client tools (EDM, CSS under evaluation, etc …) and specific high level applications written in Java, also deriving developments from the Xal framework. Relational databases are involved into the control system for equipment configuration (foreseen), machine representation and configuration, CSS archivers (under evaluation) and Irmis (mainly for process variable description). The first components of the Spiral2 control system are now used in operation within the context of the ion and deuteron sources test platforms. The paper also describes how software development and sharing is managed within the collaboration.

THE SPIRAL2 PROJECT

Ganil is one of the major facilities producing stable or rare ions beams (RIB) for nuclear physics, astrophysics and interdisciplinary research. In order to extend both the range and mass of exotic nuclei able to be provided, the Spiral2 project [1] at Ganil will deliver RIB at intensities never reached before and is built within two phases:

- The first one is the primary beam acceleration process consisting of deuterons and heavy ions sources, a warm RFQ and a superconducting linac. It also includes the S3 (Super Separator Spectrometer) and NFS (Neutrons for Science) stable experimental areas. Construction of the buildings is under way so that the installation should start in fall 2012 with the first beams expected in 2013. To anticipate the commissioning phases, the deuteron and heavy ions sources with their associated low energy beam lines are already in test respectively at CEA-IRFU (Saclay) and CNRS-LPSC (Grenoble).

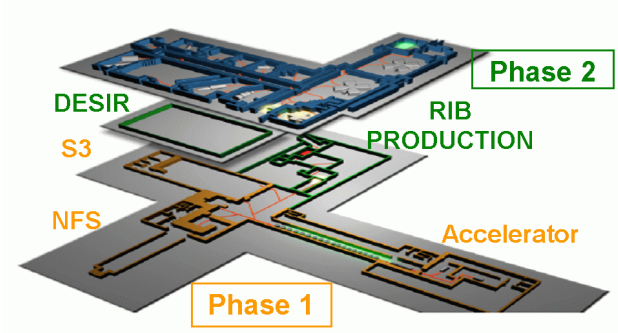


Figure 1: The Spiral2 general layout.

- Phase two concerns the RIB production complex, a new DESIR experimental room and the link to the existing Ganil facility for post acceleration and reuse of the experimental switchyard.

CONTROL SYSTEM CONTEXT

Software Development Around Epics

Since the preliminary design phase study, it has been decided to build the Spiral2 control system upon the Epics framework.

For the phase 1 control system, a collaboration between CEA-IRFU (Saclay), CNRS-IPHC (Strasbourg) and Ganil (Caen) has been set within a specific organisation so that these three laboratories share their developments [2] according to the work packages they are in charge of.

The phase 2 control system will be also based within the same Epics technical pattern, being still under design for its detailed conception. Nevertheless, it is clear that a specific point will have to be addressed as this part of the facility as it will be the recovery domain of the new Spiral2 and the legacy Ganil control systems, this last one resulting of an home-made Ada design formerly defined in the beginning of the 90's.

Main Technical Options

Most of the technical options result from the Epics environment (currently 3.14.12) which was adopted and the main choices were presented previously [3], [4].

IOC servers will be either VME crates running VxWorks (currently 6.8 release) or Linux PCs. Equipment will be addressed either directly from VME chassis or through field buses (mainly Modbus/TCP). Also Siemens S7 PLCs will be used, being in charge of specific devices or handling safety processes.

On the client side, GUIs will be, depending on the context and needs, either standard Epics tools or Java applications being developed within an adaptation to our project of the Xal framework [5].

Basic Infrastructure

In order to integrate the future needs brought by Spiral2 and to cope with the needs of the existing Ganil facility, a consequent upgrade of the Ganil central servers was performed during the yearly Ganil winter shutdown by the beginning of this year.

The new environment consists of two Dell Power Edge servers equipped with Intel Xeon E5620 processors at 2.4 GHz. SAN disks are configured as Raid10 (3 Tb) and Raid 5 (4 Tb) within iSCSI storage units. The operating system is the Red Hat Linux 6 distribution with the high availability features provided by the cluster suite package.

Shared by both the Ganil and Spiral2 control systems, this set of replicated servers is in charge of all the basic system services as well as the relational database management servers (Ingres for Ganil and Spiral2, in addition MySql envisioned for Spiral2).

Beside of these infrastructure servers, application program servers dedicated to each of the Ganil and Spiral2 machines run on specific applicative servers.

EQUIPMENT LAYER

First, linked to the equipment management (fig. 2), a development is in parallel under way for providing an environment to integrate the equipment into the control system by the end users themselves, even not Epics aware people [6]. It relies on the standard Epics template database files, the substitution mechanisms and the so-called "generation" consists of generating the starting boot file, the Epics substitutions files and (if required) the sequencer scripting files. This has been prototyped up to now within the context of quite simple equipment such as beam slits or power supplies; to be compliant with this process, developments have to respect configuration, programming rules and codification recommendations.

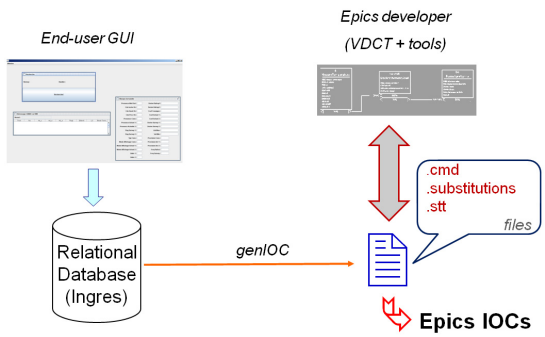


Figure 2: Equipment management work flow.

Table 1 summarizes most of the equipment to be integrated within the accelerator control system; according to the current status development, items are highlighted from green (tested during the first beam tests at Saclay and Grenoble), orange (work in progress) or red (standby).

Table 1: Equipment Integration

Equipment	Interface	Comments
Power supplies [6]	Modbus/TCP	Alarms handling still to be implemented. Various implementations according to providers.
PLCs	Modbus/TCP	Siemens S7 serie.
Faraday Cups & ACCT - DCCT Current transformers [7]	VME	Slow acquisition to get average value and fast acquisition for peak value as well as pulse display.
Slits	Modbus/RTU	Step by step motors. Temperature measurement and beam current acquisition through VME.
Fast Faraday Cup	Digital oscilloscope	Preliminary tests performed using the SCPI protocol (packet length).
Time of flight	Modbus/TCP (phase)	Specific Ganil design.
Fast current transformer	Digital oscilloscope Modbus/TCP (phase)	Preliminary tests performed using the SCPI protocol (packet length).
Transversal emittance	VME	Complex system with HT settings, Brushless motors and Faraday cups. Validated for the Low Energy Beam Transfer Lines.
Profilers	Modbus/RTU	Specific Ganil design.
Beam Position Monitors	VME64x	Specific BARC design.
Beam Losses Monitors	VME	Complex system with NIM chassis, HT modules. Specific IFIN-HH design.
Beam Energy Monitors	?	Detector not yet defined
RF amplifiers [8]	Modbus/TCP	Different for the RFQ and the Linac cavities.
LLRF control	VME64x	Specific IRFU design.
q/a=1/3 ions source [9]	VME	Complex system tested at LPSC Grenoble. Uses LabView &Epics.
Deuterons source	VME	Complex system tested at IRFU Saclay.
Timing system	Modbus/TCP	Handled by a PLC and an electronic system to tune the beam pulse shape by setting the sources, RFQ, and chopper pulsations.
Hall probe magnetic field	Modbus/TCP	Specific Ganil design.
NMR probe magnetic field	?	To be specified according to the system to be used.
Buncher, RFQ and Linac cavities	RF systems	Linked with the LLRF, RF PLCs and a private network.
Machine Protection System	?	GUIs, alarms, thresholds handling to be specified upstream the system.

Conventional facilities, cryogenic distribution and the personal access control system are out of the scope of the control system.

OPERATION INTEGRATION

Operator Interfaces and Tools

First, the EDM standard Epics editor is used for supervision applications, CSS/BOY being presently in evaluation with a specific care for its use in operation.

The archiving system is planned to be implemented with archive engines upstream a MySql database and performance tests started recently (400 values at 10 Hz monitored), jointly with the CSS client operator interface.

The alarm handling system will be a home made one (Java development), being able to process both the alarms coming from the legacy Ganil and new Spiral2 control systems. Alarms are stored in an Ingres relational database up to 330 per second for the global throughput.

Software Organisation and Standardisation

From the very beginning, to cope with the collaboration needs, a dedicated organisation was set to share developments. A Spiral2 repository was adopted both for the Epics and Java developments, including naming rules and conventions for files, directories. Also codification rules were defined for equipment.

Later, it turned out for better interactions to propose a standardized interface between IOCs and the GUI applications by defining a generic way to access equipment [10]. First, deeper codification rules were extended for Process Variable naming to handle settings, readbacks and measurements. Then, a specific database pattern consisting of dedicated records is introduced within IOC developments. It presents to any Channel Access client Process Variables to read the status word, read the default word, provide the list of all available commands to be applied, provide the list of all defaults able to be raised, get the meaning of each status bit …

High Level Applications and Machine Description

After having decided to write high level applications in Java, an investigation was done to adopt or define a framework able to fulfill our needs and requirements. Therefore, the Xal environment was evaluated within our context and, with the help of the Xal collaboration, tests were carried out successfully so that a Spiral2 derived version is now available [11]. It uses all the Xal approach and concepts and integrates some specificities coming from the new machine itself or inheriting from our previous Ganil approach.

According to the Xal schema, the facility is defined along an accelerator tree whose final leaves are the accelerator components. This description is stored in a relational Ingres database from which data are extracted to generate the Xal .xdxf compliant file. A gateway with the simulation CEA TraceWin program was added to generate theoretical values for the beam to be produced.

The first Xal based application was a general one to manage beam parameters; others under development are the beam profiles display, an optimization on-line program (using Xal algorithms), the buncher tuning and a general purpose handling tool. Many others are planned. Also standard Xal applications will be used such as the scanning programs and the magnet cycling one.

NEXT STEPS

Taking benefit of the ions and deuterons test platforms at Grenoble and Saclay, some components of the control system were tested within a quite operational context.

Nevertheless, a lot of work has still to be done with the development of the Linac control system. Also, providing more operational tools and high level applications is an issue. Lastly, the scalability of the system in a more loaded environment has to be checked carefully.

The installation is planned from summer 2012 while, in parallel, the phase 2 control system is under design.

ACKNOWLEDGEMENTS

Other contributors from the Ganil control group are: J.C. Deroy (LabView), P. Duneau and J.M. Loyant (real time), P. Lermine (databases), J.F. Rozé (PLCs infrastructure). PLCs development is managed by C. Berthe (Ganil) and R. Touzery (IRFU).

The basic hardware and software infrastructure and the network architecture are managed and provided by the Ganil Computing Infrastructure group.

REFERENCES

[1] E. Petit et al. “Progress of the Spiral2 project“, Ipac 2011.

[2] D. Touchard et al. “The Spiral2 command control software organisation and management”, Icalepcs 2009.

[3] E. Lécorché et al. “Development of the future Spiral2 control system“, Icalepcs 2009.

[4] D. Touchard et al. “Status of the Spiral2 control system“, PCaPAC 2010.

[5] C. Allen et al. “XAL Status report – Fall 2009“, Icalepcs 2009.

[6] D. Touchard, C. Haquin “A Modbus TCP based power supply interface“, PCaPAC 2008.

[7] F. Gougnaud, P. Mattei “The first steps of the beam intensity measurement of the Spiral2 injector”, this conference Icalepcs 2009.

[8] D. Touchard et al. “The Spiral2 radiofrequency command control“, this conference.

[9] F. Gougnaud et al. “The implementation of the injector Spiral2 control system“, this conference.

[10] C. Haquin et al. “Spiral2 control command: a standardized interface between high level applications and Epics IOCs”, this conference.

[11] L. Philippe et al, “Spiral2 control command: first high level applications based on the Open XAL library“, this conference.

THE COMMISSIONING OF THE CONTROL SYSTEM OF THE ACCELERATORS AND BEAMLINES AT THE ALBA SYNCHROTRON

D. Fernández-Carreiras, F. Becheri, S. Blanch, A. Camps, T. Coutinho, G. Cuní, J. Gigante, J. Klora, J. Lidón, O. Matilla, J. Metge, A. Milán, J. Moldes, R. Montaño, M. Niegowski, C. Pascual-Izarra, S. Pusó, J. Jamroz, Z. Reszela, A. Rubio, S. Rubio-Manrique, A. Ruz, CELLS, Cerdanyola del Vallès, Barcelona, Spain

Abstract

Alba [1] is a third generation synchrotron located near Barcelona in Spain. The installation of the Control System for the Accelerators (Booster, Storage Ring, Transfer lines and Front-Ends) finished at the end of 2010. The final functional tests took place during the first weeks of 2011, right before the commissioning of the Storage Ring, which started the 8th of March. Currently we are carrying out the final stages of the installation and commissioning of the seven experimental stations of the phase 1 (known as Beamlines at Alba). The control system is based on the middle layer provided by TANGO combined with open source SCADA, Sardana [2][3][4]. It is highly distributed relying extensively on Ethernet as a Field Bus. It combines diskless machines running Tango on Linux and Windows, with distributed PLCs connected to remote peripheries. In several cases, application specific hardware has been produced in house. This paper describes the design, requirements, challenges and the lessons learnt during the installation and commissioning of the control system.

THE CONTROL SYSTEM

The infrastructure implicated in controls includes more than 350 racks, 6300 equipments and 17000 cables. The controls architecture is highly distributed, comprising about 2500 network devices. The central Tango Installation for the Control and data acquisition of the accelerators has today 5680 devices and 1480 processes running.

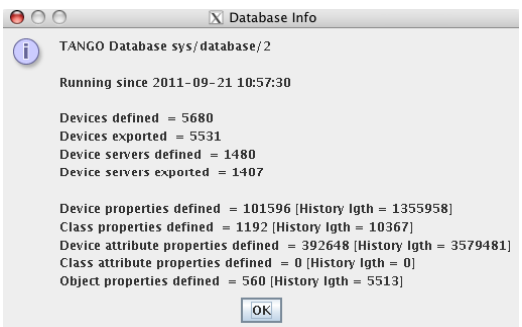


Figure 1: Summary of the Tango database for the control system of the Accelerators.

Besides the controls for the Accelerators, there is one control system per Beamline, implementing the same architecture, running on industrial PCs installed in the Beamline and virtual machines in the computing room. Each Beamline has an independent Tango installation, controls hardware and disk space in the central storage. The networks between machine controls and Beamline controls are separated, having a firewall in between. Thus the interfaces between Machine and Beamlines are implemented via virtual machines acting as gateways, and PLC direct connections.

In this first deploy of the control system for the accelerators comprises about 84 cPCI, and 22 industrial PCs distributed in the service area, all diskless running Linux openSUSE11.1 with a kernel 2.6.27 (known as Input/Output controllers or IOCs in our naming convention). The central services run in multicore servers or virtual machines in the computing room. We extensively use Nagios for monitoring controls IOCs, workstations, services and servers. Nagios combined with the Alarm system [5] has been of a great utility during the commissioning and operation.

INSTALLATION

The installation of the control system started with a prototype for the control system of the Linac-to-Booster transfer line (LTB), which was installed at the end of 2007. That installation served to test our management tools, external contractors, and to tune the project planning and execution. The year 2008 was mainly focused on the mechanical installation of the equipments of the Accelerator. In 2009 the booster was carried out and in 2010 the Storage Ring. The installation of the controls for the Beamlines was more spread in time. It has mostly occurred between 2010 and 2011.

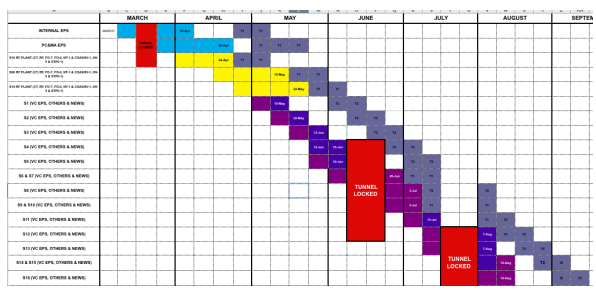


Figure 2: Cabling installation and verification, sequenced over the 16 sectors.

One of the biggest challenges was the coordination of the cabling in the different sectors combined with the check of the different subsystems, in particular EPS. The installation of the cabling for the Accelerators has been accomplished roughly in thirteen months, four months for

installing about 1200 cables of the booster and nine months for installing the 9000 of the storage ring.

MANAGEMENT TOOLS

During the installation and commissioning, a detailed plan, the commitment of many people including external contractors, and the assistance of a number software applications have been determinant for success. The most important are the cabling database [6], the request tracker (RT), the timeDB and the project management tool.

Cabling Database

A central database manages all the definitions, names and plug information related to equipments and cables. This is the central repository where types of equipments, cables and connectors are declared allowing later managing instances of every item. The types of equipment, connectors, cables and "cable configurations" are defined and inventoried as the first step. Next, instances of equipments, cables and connections between equipments are defined. Once this was completed, the cable lengths have been calculated and the manufacturing order has been determined. After the cables have been received they were installed in the final position, labelled and verified. The cable database plays a key role in all phases. Reports for manufacturing, labels, and diagnosis of the tests are generated from the database. In order to do this, the information must be consistent at any time. This has been a great challenge and an important key for the success.

Having a consistent and up-to-date central repository of the equipments and cables for the control system opens many new possibilities from the installation and operation point of view. During the pre-installation phase, the equipments were assembled, mounted and tested in the warehouse, and the record of those tests was kept in the database. Also valuable information like MAC addresses, IP addresses, boot servers, etc. was added during this phase. This made possible to create from the cabling database, configuration files for several network services, such as DNS, DHCP, Radius.

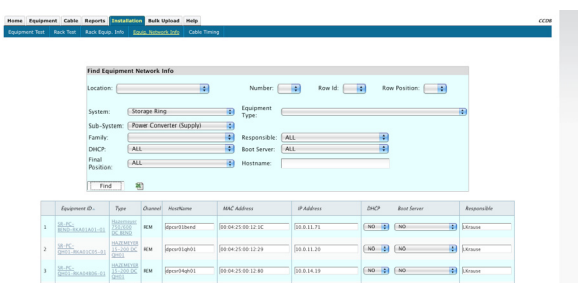


Figure 3: Snapshot of the web interface of the Equipment and Cabling database.

Many components of programs running in the PLCs of the Equipment Protection System are generated automatically from the database. The huge effort done for keeping the database up-to-date and consistent pays back here, when software components are generated automatically, reducing errors and increasing the efficiency. A Visual Basic Script running in excel creates files containing the declaration of variables, data structures, software tasks, Modbus mapping and documentation. The Logic is still programmed manually due to the difficulty managing the large number of exceptions to the standard interlock rules. In addition, Tango attribute names and the expert GUIs are also generated from the controls equipment and cabling database.

RT

RT (Best Practical Solutions LCC), has been intensively used in the computing division. It is a great tool for implementing ITIL [7] practices. Tickets are distinguished between Incidents, problems, User Requests and Request for changes. After their creation they are assigned to Unit services and to a responsible.

TimeDB and the Project Management Tool

The time spent in projects, services, problems or any other task is logged and managed by another web application, the timeDB.

Projects are managed following the PRINCE2 methodology [8], and assisted by the Computing Project Management tool, another application developed by th Management Information System group (MIS), and extensively used by the entire Computing division. It handles Business Cases, Plans, Risks, Communication management, etc. up to Project Initiation, Highlight and Exception Reports.

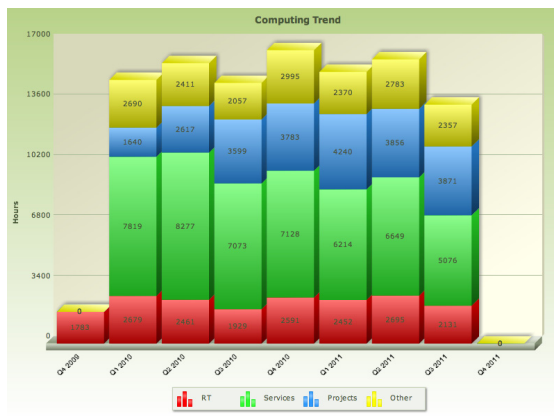


Figure 4: Trend chart of the time allocated in the computing division in the years 2010 and 2011. Green corresponds to Services, Blue to projects and Red to RT.

COMMISSIONING

The commissioning of the control system started at the beginning of 2008. A prototype of served to both test the concept and do commissioning of the LINAC and the transfer line. Later, at the end of 2009 the commissioning of the Booster started. It took place during few days in

2009 and the first weeks of 2010. At that time, RT was setup and tuned, adding the concept of services (as described in ITIL best practices), combined with incidences, problems and different type of requests. We also improved the project management tool and the timeDB.

In November 2010, the control system for the machine was ready for operation, but unfortunately, due to a delay in the authorization permit issued by the local authorities, we started only few weeks later, the eight of March 2011. In the meanwhile, we changed the schedule and we installed 3 (out of 6) insertion devices. The 16th March 2011 we had the first accumulated beam and the 18th we opened the first Front-End (BL09: a Beamline on a Bending magnet). In June the Front-Ends corresponding to the two helicoidal undulators and the conventional wiggler were open. The Machine was shutdown the tenth of June, to allow the installation of the remaining insertion devices.

During the summer we installed two in-vacuum undulators and one superconducting wiggler. We setup the cabling, electronics and the control system and we restarted the machine at the end of August.

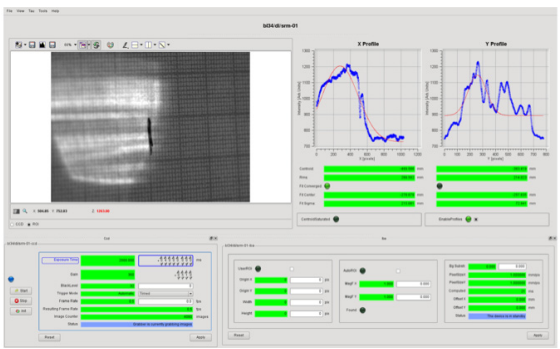


Figure 5: Snapshot of the first beam seen as visible light from a bending magnet taken the diagnostics Beamline.

During the commissioning, the management tools and the on-call service were fully operational. The computing division had a total of four persons on-call simultaneously, two controls engineers, one Electronics engineer and one system administrator. In the RT's controls queue, there were 192 standard requests and 49 tickets directed to the on-call service. These tickets were about half incidents half request for changes.

There were few looses of the beam due to the control system. Most of these unjustified interruptions of the service, have been due to Interlocks, either provoked by the Personnel Safety System (false intrusions or errors in the logic activating the beam killer) or by the Equipment Protection system (Interlock boards, water cooling, Ethernet Powerlink communications). Both are particular sensitive to any issue because they operate in many cases directly stopping the accelerators. After the first days where the issues were identified and fixed in the case of the EPS and avoided in the case of the PSS, the number unexpected interlocks was remarkably reduced. The PSS was fixed on the 23rd of June, and is already accepted and certified by TÜV.

CONCLUSION

The commissioning of the control system is an important and time-consuming task, which shall be planned, and considered accordingly in the time schedule. It is particularly tricky, because it is the last one before the commissioning of the accelerators, and it has dependencies with all the predecessors, hardware installation, equipments, fluids, sensors, cables, etc. It is necessary to break it down in parts, typically divided in subsystems, and always regarding the progress of the installation of the equipments. The equipment protection system is a particularly complex example, because it is spread all over the place, counting more than eight thousand signals installed at different times in different places. It always required three phases; checking of the cables, functional tests of the subsystem or sector and finally the general functional test.

Having a pre-installation campaign was crucial for speeding up the final installation. Most racks were preinstalled in 2007 and 2008, including equipments and internal cables, which resulted in a significant workload advanced in time.

The management tools, in particular the cabling database was crucial for maintaining the consistency and reducing the tests and early maintenance costs.

Although for a new installation like Alba, apparently operation and installation do not interfere, it was not the case. During the installation, there were quite a few "routine" operations, like Factory and Site acceptance tests of equipments, Bake-outs of vacuum chambers and equipments, etc. which implicated the different support groups in Computing, having an important impact in the final workforce for development and commissioning.

CONTRIBUTIONS

Many people have worked in this project. This is a work achieved with the effort of the whole computing division. The whole controls, electronics, network system administration groups actively participated in the design, installation and tests of the control system. Among them, the persons in charge of the network, M. Díaz, R. Escriba, had an important role. Furthermore, the help of the Management Information System group, in particular, O. Sánchez, I. Costa, A. Nardella, A. Klora, D. Salvat and V. Prat, who developed and maintained the cabling database, the TimeDB and the Project Management system has been crucial.

REFERENCES

[1] D. Fernández-Carreiras et al. "The design of the Alba Control System. A Cost-Effective Distributed Hardware and Software Architecture". ICALEPCS 2011. Grenoble. FRBHMUST01.

[2] T. Coutinho et al. “SARDANA: The software for building SCADAS in Scientific Environments”. ICALEPCS 2011. WEPMSO23

[3] http://www.tango-controls.org/static/sardana/latest/doc/html/index.html Sardana Official Documentation.

[4] http://www.tango-controls.org/static/taurus/latest/doc/html/index.html Taurus Official Documentation.

[5] Sergi Rubio-Manrique et al. “Extending the Alarm Handling in Tango”. ICALEPCS 2011. MOMMU001.

[6] D. Beltran et al. “The Alba controls and cabling database”. ICALEPCS 2009. Kobe.

[7] http://www.itil-officialsite.com/home/home.aspx ITIL. Information Technology Infrastructure Library. The APM Group Limited, Sword House, Totteridge Road, High Wycombe, Buckinghamshire, UK, HP13 6DG

[8] http://www.prince-officialsite.com/ PRINCE2. Projects IN Controlled Environments. The APM Group Limited, Sword House, Totteridge Road, High Wycombe, Buckinghamshire, UK, HP13 6DG

ISHN ION SOURCE CONTROL SYSTEM OVERVIEW

M. Eguiraun*, I. Arredondo, M. Campo, J. Feuchtwanger, G. Harper, S. Varnasseri,
ESS Bilbao Consortium, Spain
J. Jugo
University of Basque Country, Spain

Abstract

ISHN project consists on a Penning ion source, which will deliver up to 65 mA of H- beam pulsed at 50 Hz, with a diagnostics vessel for beam testing purposes. The present work summarizes the control system of this research facility. ISHN consists of several power supplies for plasma generation and beam extraction, including auxiliary equipment and several diagnostics elements. The control system implemented with LabVIEW is based on PXI systems from National Instruments, using two chassis connected through a dedicated fiber optic link between the HV platform and ground. Source operation is managed by a real time processor, while additional tasks are performed by means of an FPGA. In addition, the control system uses EPICS Archiver and Hypertable database for data logging. The integration of EPICS into the control system is done by deploying a Channel Access Server in the PXI Controller,.

INTRODUCTION

The ESS-Bilbao (ESSB) light ion linear accelerator has been conceived as a multi-purpose machine, useful as the core of a new standalone accelerator facility in southern Europe [1]. The project aims to develop significant capabilities needed to support the country participation in a good number of accelerator projects worldwide. In this context, the designed accelerator should serve as a benchmark for components and subsystems relevant for the ESS project as well as to provide experience on power accelerators science and technology.

In this context, the development of a front-end test stand for ion sources will allow ESSB to test, develop and optimize ion sources and their working parameters. Currently, a modified penning ion source on loan from ISIS is being tested [2]. This project serves several goals. It generates experimental data that can be contrasted with simulations and it serves as test stand for control systems and hardware (data acquisition, diagnostics, etc.).

A penning type ion source is operated applying a pulsed discharge periodically ($800\mu sec.$, $30A$) to source's cathode, generating a mixed plasma of Hydrogen gas and Caesium vapor. Hydrogen is delivered applying a pulsed voltage to a piezovalve for $250\mu sec.$, and its voltage is automatically controlled for maintaining a constant pressure in the plasma chamber. Once the plasma is stabilized ($\sim 500\mu sec$ later), a pulsed extraction voltage is applied ($300\mu sec.$, $12kVolt$) generating an H^- beam. This pulse diagram is repeated at $50Hz$, see Figure 1. In order to accelerate the beam, the platform high voltage power supply usually runs at $35kVolt$. In addition, a continuos power supply ($600V$ and $2A$) is needed for conditioning ion source's electrodes. This power supply is only used in the beginning of the operation. The main devices required for source operation are listed in Table 1

*meguiraun@essbilbao.org

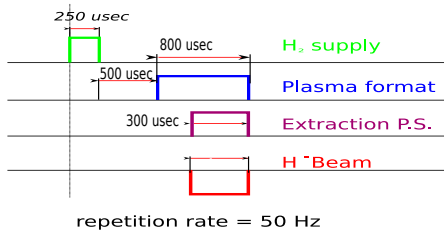


Figure 1: Pulse diagram for beam generation.

Table 1: Main Devices for Ion Source Operation.

Utilities	*Plasma creation*
Cs Heaters	DC Power Supply
H_2O Refrigeration	Pulsed Power Supply
Air Refrigeration	H_2 Piezovalve
Vacuum pumps and sensors	
Beam Extraction	
Extraction Power Supply	
High Voltage Platform Power Supply	

Once the beam is extracted, it goes through the so called diagnostics vessel, Figure 2 picture. The first diagnostics are the current transformers (ACCT and DCCT). Then, a quadrupole focuses the beam and a dipole is used for obtaining information about the degree of stripping and beam species components. The least beam diagnostics are a Faraday Cup for measurement of beam current, a Retarding Energy Analyzer for energy spread analysis, and, finally, a pepperpot device for obtaining emittance values. These devices, cannot be activated simultaneously and are also used as beam stoppers.

The control system is in charge of all of these elements, ensuring its correct operation and granting safety for both personnel and machine. In the following section, the architecture of the system will be explained, extending the information of [3].

CONTROL SYSTEM OVERVIEW

The control system architecture can be observed in Figure 3. Due to the high voltage in the platform, the main

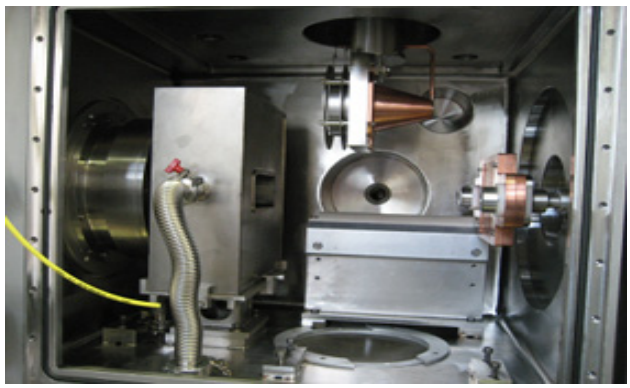


Figure 2: Diagnostic devices: from left to right, ACCT and DCCT (not visible), Quadrupole, Dipole, Faraday Cup, RPA (behind) and Pepperpot grid.

system is divided into two control subsystems, located at ground and at the platform, respectively. Each area has its own operator panel, allowing the operation of the ion source from both control desks, but not simultaneously. However, continuos monitorization and data logging of the status of the system is always active in the both panels.

The control system architecture for source operation and data acquisition is mainly based on hardware from National Instruments. Two PXI (PXI-1042Q and PXIe-1065) chassis are used, one at ground and other at platform. They are linked with a dedicated MXI fiber optic card from same vendor, and therefore, the two-chassis system can be treated as a single element from the point of view of the control designer.

The chassis located at ground contains a Real Time controller programmed in LabVIEW. It manages Cesium boilers' and pressure's control loops, some data acquisition through other PXI cards and the FPGA and communication with the operator's control desk using shared variables. It also communicates with some power supplies by Modbus/TCP and GPIB and pressure sensor by serial protocol link. In addition, the EPICS server is deployed in this device.

On the other hand, the FPGA located on the HV platform controls diverse elements. It manages the pulse signals for plasma power supplies and H_2 supply, generating the pulse diagram of Figure 1. In addition, it manages some safety signals from and to plasma power supplies and controls several electrovalves for gas supply and refrigeration of the ion source. Being the only card with signal acquisition capabilities in platform, it is in charge of acquiring voltage and current waveforms from plasma generation power supplies. Furthermore, it also measures gas flow rate, temperatures of the ion source and some other standard parameters.

As there are two operator panels, the code should avoid data overwriting between both monitorization applications. Therefore, two mode of operation are defined: local and remote.

All the code has been programmed in LabVIEW: the Real Time application, the FPGA and both monitorization programs.The main operator panel runs in a Linux environment. This is recommendable since the necessity of the EPICS archive engine.

In addition to the main controllers, a Twido PLC manages the operation of the vacuum system, which consists of two mechanical and two turbomolecular pumps. The advantage of this modular solution is to have the ability of performing stops and upgrades of the main control system without disturbing vacuum operation. This is very important in the commissioning phase, where adjustments of diverse elements and minor changes of the control system are usual. Moreover, a modular control approach leads to the possibility of reuse code and proposed control solution in other projects.

Two safety PLCs from PILZ located in both cabinets ensures safety during operation. The one located at ground handles several safety sensors (doors, grounding arm status, ...) and ensures that power supplies only receive electrical supply if safety environment is guaranteed. The safety PLC located in the platform handles the signals related to the H_2 supply and the electrical power supply of the Cs heaters. If any problem is detected or if the emergency stop is pressed, safety system shutdowns high voltage power supplies.

The fiber optic link is used for ethernet communication, and the real time controller, both monitor panels, plasma power supplies, PLCs and some other noncritical devices are attached to this network.

The following sections will briefly explain the main changes that have been implemented in the last upgrade. The most important one is the addition of an EPICS server to system, along with a new data logging system, based also in EPICS. Finally, the timing system is explained.

EPICS Integration

With additional minor improvements and modifications in current control system, an EPICS server has been implemented. It is based on the implementation by National Instruments (LabVIEW DSC module), and runs on the real time system in the PXI controller. As the first version of the program was a pure Labview implementation, some parameters of the control system are left out from EPICS server, for example some PID gains. The parameters that are published via EPICS are mainly for system configuration and machine operation, along with data acquisition variables, resulting in 65 Process Variables. The current implementation lacks of several critical capabilities of PVs such as alarm handling and extended field support. For that reason, EPICS server is used just to communication, without the full functionality that an IOC provides.

The Real Time controller publishes those PVs taking data from LabVIEW's Shared Variables, but since the main operator panel runs on Linux (there is not support for SV on Linux), this program must read and write data directly

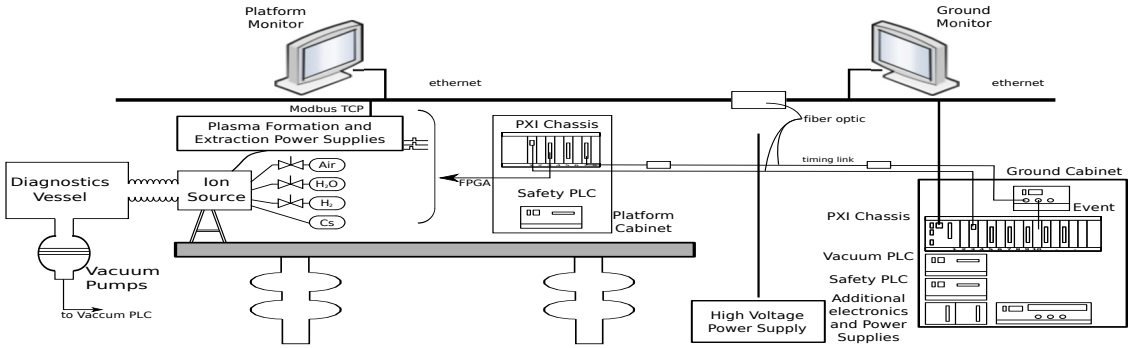


Figure 3: Control System Architecture.

from and to EPICS PVs. For that purpose CA Lab LabVIEW library is used [4]. It provides some vi-s for interfacing EPICS and LabVIEW and can be compiled for a Linux environment.

An alternative to NI approach is the one presented in [5], a pure CA server implementation in LabVIEW. It is under development but once released it will be tested to check if it fulfills the control requirements as presented in this work.

Data Archiving

As far as the database architecture is concerned, it was decided to keep the MySQL schema provided by the ArchiveEngine utility [6], an EPICS ChannelAccess client. Using this client, any channel served by any ChannelAccess server can automatically saved to a relational database, like MySQL or Oracle. In present case, Hyperachiver, a customized version of the RDB Archiver, and HyperTable, as an alternative database to MySQL or Oracle, are used. This has been developed at INFN laboratories at Legnaro (Italy) and modified in ESSB to fulfills ISHN project requirements. Since this database choice requires a linux environment, the current implementations is running on Fedora14.

PVs of array type are stored every $5sec$, the rest of the variables at $1sec$ except the ones that are recorded only on changes. This gives a total amount of data around $100MB/hour$. A graphical user interface was developed in python at ESS-Bilbao in order to have at disposal a simple data visualization tool, lighter than CSS and, as it was proved by the tests' results, faster than CSS Data Browser, which had already been used by people at INFN/LNL.

Timing System

An independent high resolution synchronization system has been designed to generate timing for triggers of plasma generation and extraction as well as data acquisition for beam diagnostics. The current design is based on commercial devices. A master event generator located at ground cabinet produces two event signals at $50Hz$. One needed in ground PXI to trigger data acquisition for beam current transformers and Faraday Cup, and the other one to trigger the generation of pulses for the plasma formation power supplies, passing through electrical-optical converters. On each chassis there is a stable and precise dedicated timing card (PXIe-6672 and PXI-6651) which receives the event signal and routes it to a PXI Trigger line. Then, a DAQ card at ground and the FPGA inside the chassis of platform starts data acquisition and pulse generation, respectively. The approximate delay of the signal due to the optical-electrical converters and the difference in cable length is around $100nsec.$, which is compensated in the event generator configuration.

FUTURE WORK

The ISNH control system is under development in various aspects, as for example the timing system. Some delay cards (Green Field technology GFT-9404) are being tested in the laboratory, but the lack of LabVEW Real Time support prevents for its usage in the facility.

In addition, full features of EPICS will be implemented, by means of a different implementation, or by deploying an extra IOC in the host computer. The most interesting features are first, the alarm handling through the so called alarm server and its operation in Control System Studio. Even if it is not required at this development stage, the EPICS gateway and nameserver are also interesting features to be implemented.

REFERENCES

[1] ESS-Bilbao home page, http://essbilbao.org

[2] D. C. Faircloth et al., "The Front End Test Stand High Performance H- Ion Source at RAL", Review of Scientific Instruments, Volume 81, Issue 2, 2010.

[3] M. Eguiraun et al., "ISHN Ion Source Control System Overview and Future Developments", Proceedings of IPAC' 11, San Sebastian, Spain Sep. 4-9, 2011.

[4] CA Lab, http://www-csr.bessy.de/control/SoftDist/CA_Lab/

[5] A. Zhukov et al, "EPICS Channel Access Implementation in LabVIEW", Proceedings of ICALEPS'11, Kobe, Japan, octuber 12-16 2009.

[6] M. del Campo et al., "EPICS HyperArchiver: Initial Tests at ESSBilbao", Proceedings of IPAC' 11, San Sebastian, Spain Sep. 4-9, 2011.

SOLARIS PROJECT STATUS AND CHALLENGES*

P. P. Goryl#, C. J. Bocchetta, K. Królas, M. Młynarczyk, R. Nietubyć, M. J. Stankiewicz, P. Tracz, Ł. Walczak, A. I. Wawrzyniak, National Synchrotron Radiation Centre Solaris at the Jagiellonian University, Krakow, Poland
K. Larsson, D. Spruce, MAX IV Laboratories, Lund, Sweden

Abstract

The Polish synchrotron radiation facility, Solaris, is being built in Krakow. The project is strongly linked to the MAX-IV project and the 1.5 GeV storage ring. An overview will be given of activities and of the control system and will outline the similarities and differences between the two machines.

INTRODUCTION

The first synchrotron light source in Poland will be built in Krakow. It will be located at the III Campus of the Jagiellonian University. The project is hosted by the Jagiellonian University and funded with EU regional development funds of 143 MPLN. The contract for the establishment of the National Synchrotron Radiation Centre Solaris was signed between the Jagiellonian University and the Polish State on March 9th, 2010.

MAX IV Collaboration

The agreement between the Jagiellonian University and Lund University in Sweden is crucial for the success of the Solaris project. It was signed in November, 2010 following earlier Memorandum of Understanding. According to the agreement between these institutions, the Solaris facility will be built based on the MAX IV design. The project is based on a close cooperation with the MAX IV project. MAX IV provides not only design but also expertise, support and training for the Solaris team.

Machine Description

Machine will consist of a 1.5 GeV ring and a 550 MeV linac injector (see Fig. 1). The ring will be an exact copy of the smaller ring of the MAX IV project but with differences in beamlines, insertion devices and general infra-structure.

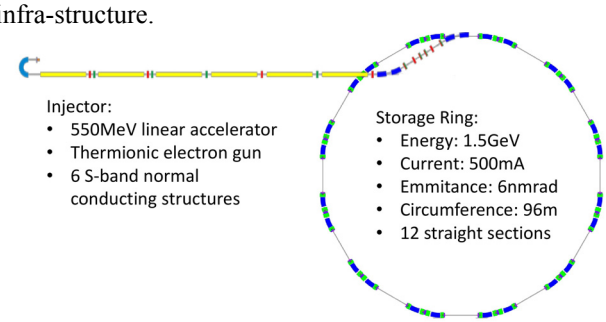


Figure 1: Layout of the machine.

* Work supported by the European Regional Development Fund within the frame of the Innovative Economy Operational Program: POIG.02.01.00-12-213/09
#piotr.goryl@uj.edu.pl

A thermionic electron gun will be used as the source of electrons. The beam will then be accelerated by an S-band normal conducting linac. The linac injector will be built with the same components of MAX IV incorporating similar design concepts. Since at the first stage of solaris the linac will not provide full energy injection, a ramping procedure will be implemented. Infra-structure and the design of the accelerators will permit a future implementation of top-up [2].

CURRENT STATUS AND ACTIVITIES

The status of Solaris has been reported in [1]. In this paper the control system status and IT related issues will be emphasized.

Since the machine will be a replica of the MAX IV 1.5 GeV ring it is reasonable to have the same control system solutions. Therefore most of the control system concepts for Solaris follow relevant ones for MAX IV.

The TANGO [4] control system and its dedicated tools have been chosen for the integration layer. TANGO has been implemented at several laboratories in Europe and found to be both reliable and mature. Close collaboration between laboratories involved in TANGO projects provides plenty of software tools, components and support.

From the tools the TANGO community provides several have been chosen. As an example Sardana developed at ALBA will be used for sequencing purposes. Graphic user interfaces will be built with different tools like Taurus, Jddd, JDraw and the ATK library. Among them Taurus and Python scripting language is preferred.

Development

Subversion and the Redmine server for a source code sharing and projects' tracking have been adopted.

There are several on-going software developments at MAX-lab. Some of them are directly connected to the new facility. Others are for maintaining the current laboratory in operation. All new development at MAX-lab, however, is using the TANGO control system and may be regarded as preparation and proof of the system.

Developments related to future control system include: the pre-injector test stand control system, the Virtual Accelerator and the Allan Bradley PLC device server.

MAX IV is cooperating with Soleil to adopt their motion control standard and evaluate a new standard in the future.

Evaluation

Many of activities are on going to find and evaluate solutions to be implemented for both MAX IV and Solaris. The MAX III ring has been integrated into TANGO using channel access device servers. TANGO historical and temporary archiving is in operation at MAX-lab for more than one year and several beamlines are using TANGO at the facility.

Network Reflective Memory based on the FERMI@Elettra solution is being evaluated and adjusted in cooperation with Fermi [3]. It is likely that MAX IV will use the image acquisition and processing system developed at Fermi too.

Procurements

There are several procurements at MAX IV related to the control system. Recently, two framework agreements have been awarded:

- For control system services and support
- For PLC's hardware and support

All of them include the option for Solaris. According to the agreement between the Jagiellonian and Lund Universities, Solaris may purchase specific hardware or services using a single source procurement procedure to ensure compatibility amongst the two machines.

In parallel to the above activities, the construction of the building infra-structure is on-going in Krakow. It includes power, water and heating, air conditioning, the security system and computer network.

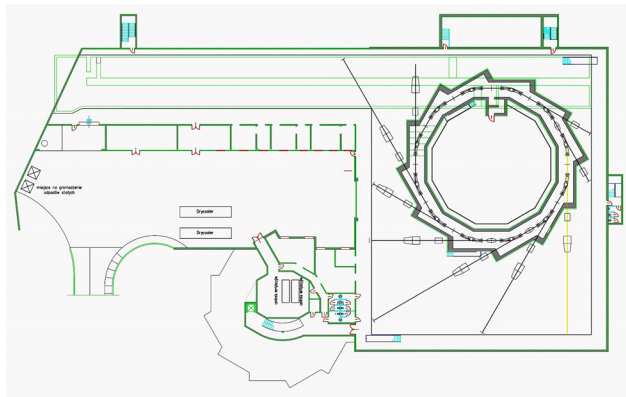


Figure 2: Concept layout of the facility.

Network

The computer network will be connected to the Jagiellonian University campus network. However, it will be operated independently from the rest of University network at least for the accelerator control systems. The general layout has already been designed along with some technological solutions to be used. Separation between the control network, the office network and the campus network will be implemented with VLANs and an on-site firewall. Additionally it is planned to use stackable switches in a main distribution point with the office network and control system network connected to separate similar units (see Fig. 3).

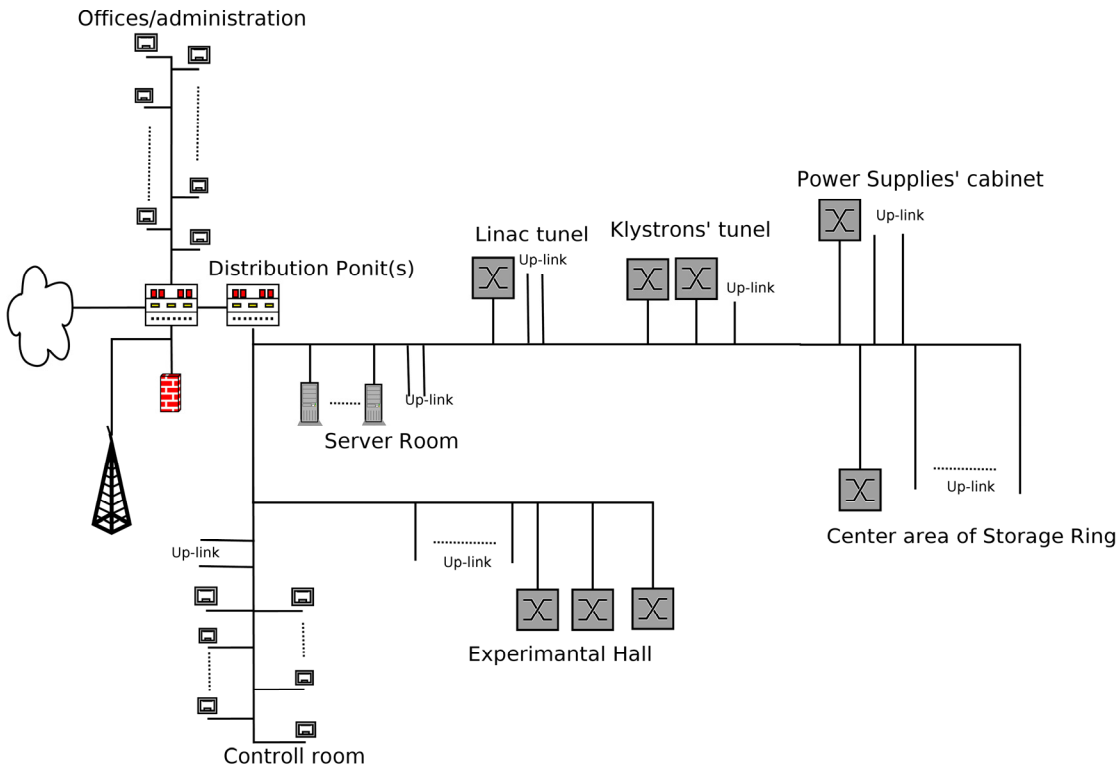


Figure 3: Concept network diagram.

Offices will be equipped with a 1GbE network whereas the accelerator control system will be provided with 10GbE fibre access points in specified locations. At these locations switches will be installed according to needs. Redundancy will be implemented through the provision of backup uplink connections between neighbouring switches, assuming a spanning tree protocol will be used.

This has been found to be a cost effective solution and flexible enough for future upgrades. Compatibility with Network Reflective Memory should be achieved, too.

DIFFERENCES TO MAX IV

In general, the Solaris control system will be identical to MAX IV. The main goal is to have both systems as similar as possible. If there is any dissimilarity at least the same tools (software, protocols etc.) should be used.

Differences between Solaris and MAX IV come mainly from project physical and logical dimensions (scale) and location. The scale of the Solaris project means there is no need to implement all the features of the MAX IV control system. The Solaris team will handle location and environmental differences.

Most of the work resulting from these differences may be related to configuration but it depends on a design of affected element. Configuration may be regarded as a part of installation and not as additional work. However, it may require pre steps like re-designing.

Differences Between Solaris and MAX IV

Listed below are the identified differences grouped by source and in brackets the consequent work.

Location:

- Network (design, configuration)
- Server and equipment location (installation)
- Cable trays (installation)
- Racks (installation)
- Geographical GUIs (design and development)

Equipment differences:

- Beamline (design, development, configuration, installation)
- EPS (configuration, re-development)
- Standard servers and workstation hardware (no impact)

Project scale:

- One mode linac operation
  - Triggering system (configuration)
  - GUIs (simplification, configuration, development)
- No need for Network Reflective Memory or equivalent
- Shorter linac, no bunch compressor
  - GUIs (simplification, configuration, development)
- Ramping (development, configuration)
  - GUIs (configuration, design, development)
  - Operation logic (development, configuration)
- Number of signals
  - Less archiving space
  - Less network load

Environment:

- Staff size (management)
- Office infrastructure and services
- PSS (development, purchase)
- Schedule (management)

Schedule Impacts

MAX IV and Solaris schedules are compatible in terms of purchasing and installation. Commissioning of the Solaris ring will, however, occur before that of the MAX IV rings. This means that Solaris should actively participate and follow work in Lund in preparation of a rapid systems start-up. In the area of the control system this is especially important in regard to eventual debugging of the system during the commissioning phase.

Collaboration

Collaborations with other institutes will be established to cope with differences and to handle additional work. In this respect Memorandums of Understanding with Elettra and ALBA have already been signed. This will provide additional expertise and support too.

CONCLUSION

As outlined in the paper, the Solaris project is progressing. Due to the linking to the MAX IV project the most demanding part of activities is in management. Almost all design and development is being done in Lund by MAX IV team with support being provided by the Solaris group. Work is being undertaken to design, develop and put into operation the Solaris facility and its control system. Solaris will reply on tight cooperation between the Jagiellonian and Lund Universities and on collaboration within the TANGO community. The facility will be built on limited resources and within a short time schedule through shared knowledge, shared expertise, shared workload and rapid training .

REFERENCES

[1] C.J. Bocchetta et al., Project Status of the Polish Synchrotron Radiation Facility Solaris, IPAC'11 THPC054 (2011)

[2] A. I. Wawrzyniak et al., Injector Layout and Beam Injection Into Solaris, IPAC'11 THPC123 (2011)

[3] L. Pivetta et al., The FERMI@Elettra Distributed Real-time Framework, THDAUST03, this conference

[4] http://www.tango-controls.org

THE DIAMOND CONTROL SYSTEM: FIVE YEARS OF OPERATIONS

M. T. Heron, on behalf of the Diamond Control Systems Group, Diamond Light Source, Oxfordshire, UK.

Abstract

Commissioning of the Diamond Light Source accelerators began in the 2005, with routine operation of the storage ring commencing in 2006, and photon beamline operation in January 2007. Since then, the Diamond control system has provided a single interface and abstraction to (nearly) all the equipment required to operate the accelerators and beamlines. It now supports the three accelerators and a suite of twenty photon beamlines and experiment stations. This paper presents an analysis of the operation of the control system and further considers the developments that have taken place in the light of operational experience over this period.

INTRODUCTION

Diamond Light Source [1], is a third-generation 3 GeV synchrotron light source based on a 24-cell double-bend achromatic lattice of 561m circumference. The photon output is optimised for high brightness from undulators and high flux from multi-pole wigglers.

Storage Ring (SR) commissioning was completed in two phases during 2006. An initial phase of commissioning, to establish accumulated beam at 700 MeV in the SR, took place during April and May 2006. This was followed by installation of the insertion devices for the first eight beamlines and a second phase of commissioning to establish operational conditions and undertake photon beamline commissioning. Routine user operation for the first seven photon beamlines began in January 2007.

Since January 2007 there has been an aggressive programming of photon beamline design, installation and commissioning, such that as of September 2011 there are twenty beamlines in operation, four beamlines in advanced construction or commissioning, and eight in various stages of design. During this period there have also been a number of developments of the accelerators to improve the performance and operational reliability. The operational photon beamlines have been developed with increased automation of samples and of data analysis, and the use of new detectors.

All of the above developments have required ongoing development and support from the control system. In addition, the control system has been managed to improve reliability and to prevent the onset of obsolescence.

RECAP ON THE DIAMOND CONTROL SYSTEM

The Diamond Control System [2] uses the EPICS toolkit [3] and provides a high degree of integration of the underlying technical systems. These include all power converters, most diagnostics, vacuum systems, the machine protection system, insertion devices, RF amplifiers, girder alignment, front-ends, photon beamlines, experiment stations, instrumentation and detectors.

Most of this equipment is interfaced through a range of generic VME I/O based on VME IP carriers, IP modules, transition cards and plant interface modules. For motion control, the OMS VME58 is used for straightforward applications; however, for synchronous control, the Delta Tau PMAC controller is used. Programmable Logic Controllers (PLCs) from Omron are used for interlocking and control, e.g. for vacuum valves, whilst for high-end process control applications, such as the Linac, the RF cavities and the cryo-plant controls, the Siemens S7 series of PLCs are used. The control system interfaces to the accelerator technical systems at 32 control and instrumentation areas (CIAs) and to photon beamlines at one or two CIA(s) for each beamline. The CIAs are air-conditioned rooms that maintain a clean and temperature-stabilised environment for the instrumentation.

A fibre optic infrastructure is installed from each of the CIAs back to the Control System Computer Room and from there to the Control Room. The fibres provide two computer networks, a control system network and a secondary computer network, and are also used for the timing systems, for the Machine Protection System and for the beam position feedback system.

A global Machine Protection System (MPS) [4] manages global equipment protection for vacuum vessels and series-connected magnets (SR dipoles, booster dipoles and booster quadrupoles). Whilst water and temperature are managed using PLC-based solutions local to each cell, some of the global interlocks, notably mis-steered beam, require a fast response and are therefore managed by bespoke MPS modules. The timing system uses a central event generator, with receiver modules located in each EPICS IOC.

Development and operational client platforms for the control system are PCs running RedHat Enterprise Linux.

Diamond has standardised on a combination of EPICS and GDA [5] for the control of photon beamlines, experimental stations and detectors. EPICS provides the low-level interface to the hardware and user interface functionality for engineering type operations, whilst GDA provides the science-based interface for the station scientists and visiting users of Diamond. The exception to this is where commercial instrumentation forms the basis of the experimental station.

CONTROL SYSTEM RELIABILITY

From day one of operation of Diamond, every beam loss has been recorded and analysed, along with non-

beam-loss faults, to understand failures. During the period 2007 to 2011 the number of operating hours per year has increased from 3048 hrs to 4848 hrs and the MTBF for the year has increased from 10 hrs to 53 hrs. Over this period, the number of beam loss faults caused by the control system (software, hardware and machine protection system induced) has fallen, as shown in Figure 1.

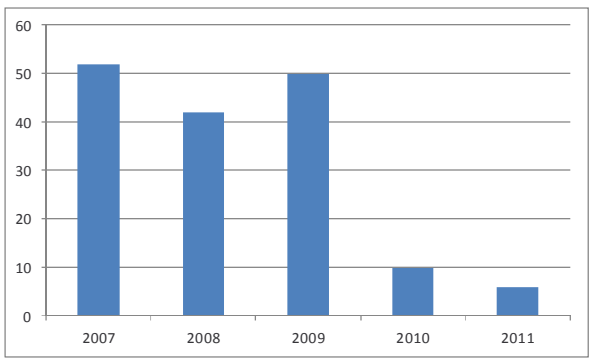


Figure 1: Number of control system faults by year to dump stored beam. For 2011, only 3648 hours have been operated out of 4848 scheduled hours.

The control system faults in 2007 were caused by a series of MPS faults related to the MPS logic for front ends, enabling beamline faults to propagate through the front ends and so causing the electron beam to be dumped. The requirements for the protection were reviewed and new logic applied to minimise these trips. This highlighted the need to take input from operations into account in the hazard analysis and assessment process.

In 2008 the faults were dominated by a series of faults in the global MPS system. These were difficult to diagnose as the beam loss causes all electron beam position monitors (BPMs) to indicate an orbit excursion interlock, within the sample time of the clock latching on the first interlock. The cause of the MPS fault was diagnosed to be a failure of optical transmitters and insufficient optical budget. In the first place the local MPS module transmitters were modified, and subsequently the global MPS module was redesigned to provide a post mortem capability. This problem highlighted the difficulties in diagnosing faults which are effectively in a feedback loop with the beam system and are beam critical.

In 2009 the faults were dominated by a failure of the Fast Orbit Feedback (FOFB), which despite traps in the software to shutdown the FOFB gracefully, still dumped the beam. It was again an intermittent fault and was due to a fault on the VME back plane on the bus grant acknowledge chain in one of the 24 VME crate controllers used for orbit feedback (possibly as a result of a tin whisker). This was again difficult to diagnose as it was intermittent and in the feedback so that it dumped the electron beam.

Whilst the underlying EPICS base software is very stable and the large deployed base quickly highlights any significant problems in new releases, each control system also includes many in-house-developed components (device drivers, database and configuration). Work has been performed on a number of tools to automate the building of EPICS databases [6] thereby minimising configuration errors, and on building unit tests for all in-house support modules. However, so far the unit tests have only been developed for a fraction of the total number of modules used.

MAJOR DEVELOPMENTS SINCE FIRST OPERATIONS

Since first operation, Diamond has installed fifteen photon beamlines, experimental stations and associated front ends. These have required in-vacuum, ex-vacuum, and cryo-cooled permanent-magnet insertion devices and two superconducting multi-pole wigglers. These have required two significant lattice modifications [7] for routine operation, adding additional quadrupoles to the SR, and a new lattice tuning for low-alpha operation to deliver electron bunch lengths of a few picoseconds. All of these have been supported by the control system. The following presents some key developments of the accelerator and beamlines involving the control system.

Control System

The machine control system was predominantly deployed with EPICS 3.13.9, (3.14.6 for Libera and soft simulations) as this was the stable version of EPICS in 2003 and so provided a stable platform for in-house work and for external suppliers delivering turn-key systems. The beamline control systems which started in development in 2005 adopted EPICS version 3.14.8.2. During the period 2008 to 2011, the machine control system was upgraded to 3.14.8.2. The difference in process and environment meant that the changes were involved and so required considerable testing. All the control systems are now in the process of being upgraded to EPICS version 3.14.11, and the policy is to try and keep all systems on a common version of EPICS base, core components and modules.

All development and operational client systems are Linux-based, whilst servers predominantly run under Linux or VxWorks, with a few Windows systems where only Windows drivers are available for a given piece of hardware. The version of Linux has been steadily upgraded such that it is currently Enterprise Linux 5, with a move to Enterprise 6 and 64 bit platforms in progress. In order to ensure quality of deployment, all applications are built on a build- and deployment-server, thereby ensuring consistency of the tool chain for the build of all operational systems.

Fast Orbit Feed Back

Diamond commenced operation running a client application to correct the orbit manually, after injection,

then once per second during stored beam. This addressed issues from long term drift but did not suppress a range of disturbance induced from changing ID gap or from the environment. During 2007 the Diamond FOFB [8] was commissioned. It takes the data, at a 10 kHz update rate, from the 172 electron BPMs to 24 VME processor cards, which carry out the feedback processing to update the 344 corrector PSUs. It uses a low-latency high-bandwidth data path which is realised by connecting the Libera units and the computation nodes using the high-speed serial interconnections available on the Xilinx FPGAs. The interconnection is realised over a fibre optic link using the Diamond Communication Controller designed and implemented in VHDL. It provides excellent rejection of disturbances from ID scanning and environmental effects up to ~100Hz.

The data path further feeds an on-line spectral analysis application, and since 2010 this data is being recorded in a short-term archiver [9], storing the most recent three days of data to enable subsequent analysis of reported disturbances. Measurements of the forward and reflected powers and phases into each of the RF cavities have also been included by using the Libera electronics as RF power meters, and from that data, disturbances originating from the RF amplifiers were identified. Current work has included the design of an interface from the communication network to photon BPMs, to incorporate BPM data into the fast archive.

Whilst FOFB achieves the desired electron beam stability of better than 10% in beam dimensions, it is foreseen that improvements in this performance are likely to be required in the future. In preparation for this, the booster synchrotron is being operated as a storage ring at 100 MeV to develop and test new control algorithms for improved FOFB operation on the SR [10].

Top-up Operation

The initial design of Diamond made provision for top-up operation in the accelerator technical systems including the control system and Personnel Safety System (PSS). As part of the implementation of top-up [11], a hazard analysis was undertaken to define a safe operating regime. This defined limits on the integrated radiation losses and transfer efficiencies, and PSS logic for injection with shutters open, such that excursions outside these constraints would cause the electron beam to be dumped by the PSS. Critical parameters must stay within the safe operating window to avoid unnecessary beam dumps and ensure reliable operation. When a problem develops, it must therefore be detected before the PSS dumps the beam, causing machine operation to drop back to decay mode and so preventing beam loss. This supervision of top-up is managed by a client application implemented in Python and Qt which ensures that these requirements are met. It further generates and compiles statistics on the operation of the technical systems required, e.g. transfer efficiencies through each of the accelerators. These are produced for the last shot, the last 10 shots, the last hour, the last six hours and the last 24 hours and so give visibility to developing problems.

Top-up operation also required the provision of additional timing signals to each of the experimental stations, as both hardware signals and EPICS PVs, to gate data acquisition during the injection when the photon beam is disturbed by the effect of the injection kickers on the stored beam.

Archiving

Diamond has tried to archive as much information as possible about the environment, accelerators and beamlines. This is valuable in diagnosing beam losses and also in understanding and characterising long-term behaviour. By 2010 there were 12.5 TB of data on line, and the system was reaching the storage capacity; further, the storage rate was approaching the limits of the hardware, and the retrieval time had become unacceptable for both very wide (in terms of number of process variables) queries and for very deep (going back in time) queries. Despite segregation of indices by technical area, the 2 GB index limit was being reached. To address these problems, new hardware consisting of 36 TB of disk in a RAID 5 configuration has been installed. The 2 GB index limit has been eliminated from the code by moving from 32-bit file addressing to 64-bit, allowing a single master index to be implemented. The XMLRPC query interface has been extended to return structured arrays of data in one query. However, it is recognised that this solution is still close to what is achievable in terms of performance from the current architecture [12]. Increases in the amount of data archived or further improvements in retrieval times will require a new storage architecture.

Motion Control

Motion is realised through a variety of motion controllers which all use the EPICS Motor record to give a common interface. As part of this we have developed a system for co-ordinated motion over multiple axes [13]. Traditionally a sample has been scanned in a stop-start way, with acceleration time limiting the rate of acquisition. To address this, trajectory scanning has been developed, whereby complex motion is pre-programmed in to the motion controller, and a hardware trigger is used to take the data on the fly. The result has been a reduction in the time taken by a data scan for EXAFS from 50 minutes to 60 seconds [14].

Video Cameras

Support for video cameras was initially standardised on a range of cameras using the Firewire interface. These were interfaced with PMC Firewire interfaces located on VME processor boards and a commercial, (binary only) Firewire stack running under VxWorks. With limited numbers of cameras per bus and limited bus length these worked reasonably reliably, but the interface failed to support the specified maximum number of cameras per bus, and so it was migrated to use PCs and Linux. Again this failed to support reliably multiple cameras per bus.

Only on detailed investigation did it become clear that the Firewire stacks under both VxWorks and Linux were not full implementations of the standard. The most complete implementation tested was under that running under Windows XP. With Firewire declining in popularity, cameras are now being interfaced using Ethernet and the GigE protocol [15].

Excalibur Detector

A 2D position-sensitive detector is currently being developed for a range of imaging experiments. It is based on Medipix3 device, a 256 x 256 pixel photon-counting detector. The hardware to interface an array of these devices is being developed jointly with STFC [16]. The detector control, calibration and readout will use EPICS and the EPICS area detector module.

Transverse Multibunch Feedback (TMBF)

To damp coupled-bunch instabilities up to 250 MHz, a TMBF system was developed [17]. It consists of a detector in the form of a set of pick-up buttons, an in-house-developed RF front end, a commercial digital FPGA- based feedback electronic module called Libera Bunch-By-Bunch from Instrumentation Technologies, and stripline kickers. The FPGA functionality and the EPICS interface were developed in house. It provides good suppression of horizontal and vertical instabilities at 250mA beam currents with zero chromaticity, together with a range of operational diagnostic functionalities including continuous measurements of the betatron tune and chromaticity

Future Interface Standards

With a third phase of ten photon beamlines approved in 2010 it was considered timely to review the choice of control system interface platform. A decision was taken to move away from VME and to use network-attached I/O for all new beamlines and photon front ends. The resulting solution uses Ethernet-attached motion controllers, PLCs, cameras and serial devices through terminal servers, and ADCs, DACs and Digital I/O through Ethernet and the Ethercat protocol [18]. The IOC servers are 1U PCs running Linux with a real-time extension, where required.

CONCLUSION

The Diamond control system has met its initial objectives and gone beyond them, particularly in the control of experiment stations, and detectors. The overall contribution to beam losses has been at an acceptable level and early issues contributing to unreliability have been identified and addressed. Nevertheless, control system reliability is an area that continues to receive attention.

However, there remain areas for further improvement, including documentation of the installed system and in provision of unit tests for all mo dules. Both of these activities are limited by the available effort and priority. The latter is likely to be of significant value when a major change in the EPICS base software components is made.

Looking to the immediate future, most of the control system hardware is now six years old. Whilst it is not showing signs of increased failure rates, a number of hardware systems have been subject to end-of-life notifications by manufacturers. In all the cases to date, this obsolescence has been readily managed. Nevertheless, all areas need to be continually assessed such that the risk level from obsolescence presented to the overall project is limited to an acceptable level..

The main EPICS client applications (EDM, Alarm handler) are X11- and Motif-based, and somewhat dated, and so a move away from these applications is now being considered.

REFERENCES

[1] R. P. Walker, “Commissioning and Status of the Diamond Storage Ring”, APAC 2007, Indore, India
[2] M. T. Heron, “Implementation, Commissioning and Current Status of the Diamond Light Source Control System”, ICALEPCS 2007, Knoxville, USA
[3] http://www.aps.anl.gov/epics
[4] M. T. Heron, “The Diamond Machine Protection System”, this conference
[5] P. Gibbons, “GDA and EPICS working in unison for science driven data acquisition and control at Diamond Light Source”, this conference
[6] M. G. Abbott, “An EPICS IOC Builder”, this conference
[7] B. Singh, “Implementation of Double Mini-beta Optics at the Diamond Light Source”,
[8] M. G. Abbott, “Performance and Future Development of the Diamond fast Orbit Feedback System”, EPAC 2008, Genoa, Italy
[9] M. G. Abbott, “A New Fast Data Logger and Viewer at Diamond: the FA Archiver”, this conference
[10] S. Gayadeen, “Diamond Light Source Booster Fast Orbit Feedback System”, this conference
[11] R. P. Walker, “Preparation for Top-up Operation at Diamond”, EPAC 2008, Genoa, Italy
[12] J. Rowland, “Algorithms and Data Structures for the EPICS Channel Archiver”, this conference
[13] M. Pearson, “Recent Developments in Synchronised Motion Control at Diamond Light Source”, this conference
[14] R. Wooliscroft, “Quick EXAFS Experiments Using A New GDA E clipse RCP GUI With EPICS Hardware Control”, this conference
[15] T. Cobb, “Integrating Gigabit Ethernet cameras into EPICS at Diamond Light Source “, this conference
[16] J Thompson, “Controlling the Excalibur Detector”, this conference
[17] I. Uzun, “Operational Status of the transverse Multibunch Feedback at Diamond”, this conference
[18] R. Mercado, “Integrating EtherCAT Based Remote IO into EPICS at Diamond”, this conference

THE DESIGN STATUS OF CSNS EXPERIMENTAL CONTROL SYSTEM*

Jian Zhuang[1,2,3], Yuanping Chu[1,3], Libin Ding[1], Lei Hu[1], Dapeng Jin[#1,3], Jiajie Li[1], Yali Liu[1], Yuqian Liu[1,3], Yinhong Zhang[1,3], Zhuoyu Zhang[1,3], Kejun Zhu[1,3]
Institute of High Energy Physics[1], Beijing 100049, P.R.China
Graduate University of Chinese Academy of Sciences[2], Beijing, P.R.China
State Key Laboratory of Particle Detection and Electronics[3], Beijing, P.R.China

Abstract

To meet the increasing demand from user community, China decided to build a world-class spallation neutron source, called CSNS(China Spallation Neutron Source). It can provide users a neutron scattering platform with high flux, wide wavelength range and high efficiency. CSNS construction is expected to start in 2011 and will last 6.5 years. The control system of CSNS is divided into accelerator control system and experimental control system. CSNS Experimental Control System is based on EPICS architecture, offering device operation and device debug interface, communication between devices, environment monitor, machine and personnel protection, interface to accelerator control system, overall system monitor and database service. The control system is divided into 4 parts, such as front control layer, local and global control layer based on EPICS, database and network service and the others. The front control layer is based on YOKOGAWA PLC and other controllers. EPICS includes local and global control layer provides all system control and information exchange. Embedded PLC YOKOGAWA RP61 and others is to be used as communication node between front layer and EPICS. Database service provides system configuration and historical data. From the experience of BESIII, MySQL is an option. The system will be developed in Dongguan, Guangdong province and Beijing. So VPN will be used to help development. Now, total 9 persons are working on this system.

INTRODUCTION TO CONTROL SYSTEM OF TARGET AND INSTRUMENTS

Neutron scattering becomes a more and more important method probe the structure of the microscopic world. In physics, chemistry, biology, life science, material science, new energy, as well as in other applications, Neutron scattering is the widely used as a complementary way to X-ray in advance research.

To meet the increasing demand from user community, China decided to build a world-class spallation neutron source, called CSNS(China Spallation Neutron Source) [1] [2]. It can provide users a neutron scattering platform with high flux, wide wavelength range and high efficiency. CSNS construction is expected to start in 2011 and will last 6.5 years.

The control system of CSNS is divided into accelerator control system and experimental control system. This paper introduces the design and some test work of the experimental control system.

OVERALL TASKS AND SYSTEM ARCHITECTURE

The main tasks of experimental control system includes: providing the global communication platform for the whole target station and instruments; providing global monitoring and database service; providing TPS(Target station Protection System); providing interface to the front end controls ; fan-out of T0 related signals to where needed; interface with the accelerator control system with their cooperation; coordinating the PPS work for the target station and instruments; local control tasks as required.

CSNS Experimental Control System is divided into global control layer (GCL), local control layer (LCL) and front end layer (FEL), as shown in Figure 1. Global control layer and local control layer are based on EPICS.

In Global control layer, network and database service are main task. Device control and TPS are dominated works for front control layer. Sub-system control is the main task of local control layer. All these are integrated in EPICS system.

KEY ISSUES AND TECHNICAL DIFFICULTIES

In Target and Instruments control system, some difficulties and key issues are focused on.

Stability and Reliability of the Key Path and Devices

Stability and reliability is the most important issue of control system. Control on key path is related to the whole system safety and availability.

To achieve this, system self-check is must. All sub system is required to have self-check ability, and the result of self check can be displayed on the central control room. All the status must be stored in database for historical display and trouble shooting in the future.

*Work supported by Institute of High Energy of Physics
Corresponding author: jindp@mail.ihep.ac.cn

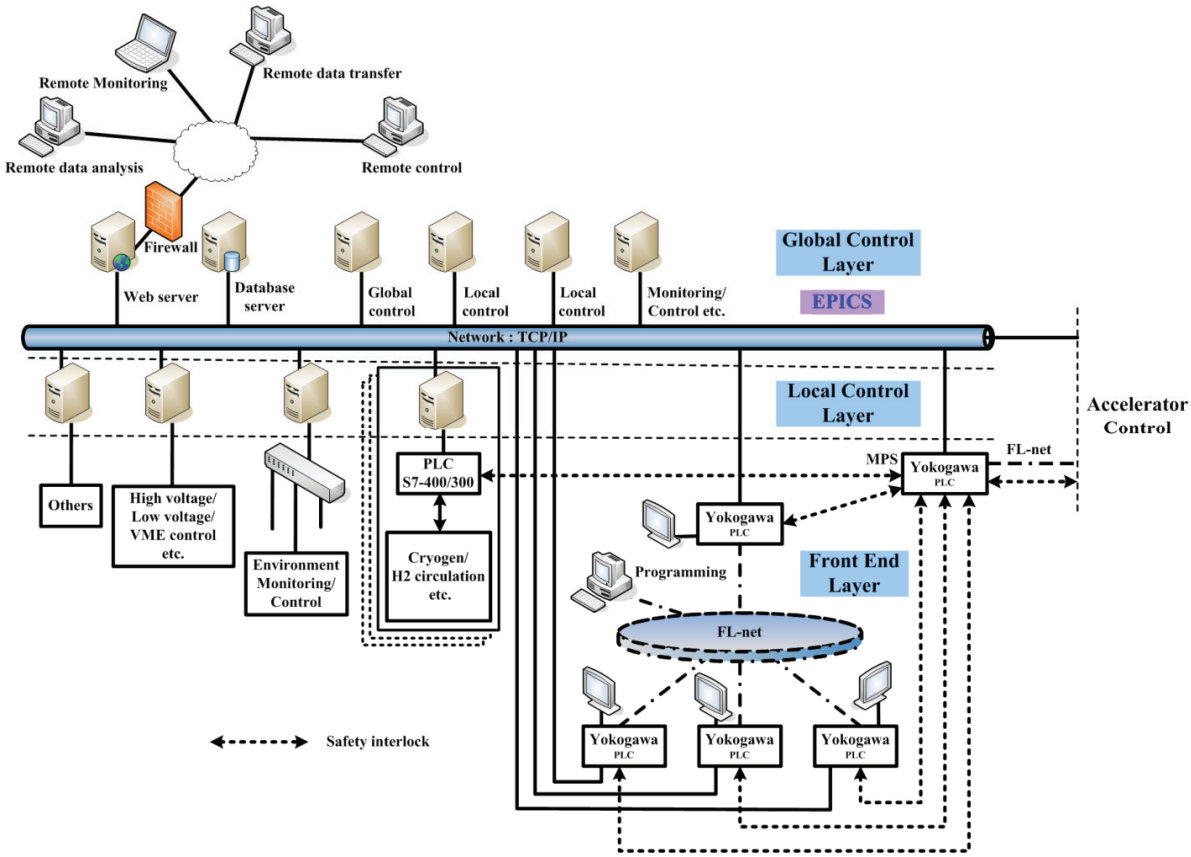


Figure 1: System Architecture.

System Scalability, Maintainability and Configurability

The experimental control system will run for several decades. For long term running, system scalability, maintainability and configurability are main issues that we should be focused on.

Budget

For any system, budget is always an important issue.

Interface with the Front End Control

In instruments, there are always many types of devices for different purpose. So, how to integrate all these devices to the whole control system is a big issue for us.

DESIGN AND TEST STATUS

Epics

CSNS experimental control system is integrated by EPICS control system. To guarantee the Stability and reliability of the system, software version control should be adopted. A software package called CCSP(Control Core Software Package) is created. CCSP includes some selected software, such as EPICS base 3.14.12, Control System Studio 3.0, and some other software. All these software are tested in the function test system to avoid bugs in software version mismatching as shown in Figure 3.

Three kind of IOC is considered to be used in CSNS experimental control system.

The embedded PLC F3RP61 from YOKOGAWA will be used as information exchange and control node in local control layer, between global and front control layer. A lot of test is done on F3RP61. Its performance and role in system is clear.

ARM based IOC will be used in environment monitoring for its low cost as shown in Figure 2. EPICS IOC test is done on a low cost ARM board based on S3C2440 CPU. The results show that illustrates the net performance of this kind board is limited. Graphical interface Qt is developed on this board. This IOC may be used in some device control, and information display.

Blade server or industrial PC is considered to be used as soft IOC to integrate other front layer device into system. Blade server can provide some virtual computers by virtualization software to achieve high reliability, availability and scalability. Soft IOC in virtual computers can provide high reliability data exchange between device and global layer.

Figure 2: IOC based on ARM.

A CA Gateway computer will be placed between experimental control net and accelerator control net to avoid net broadcast storm and increase security of whole system. The test of CA Gateway is also done in function test system, and the policy is decided.

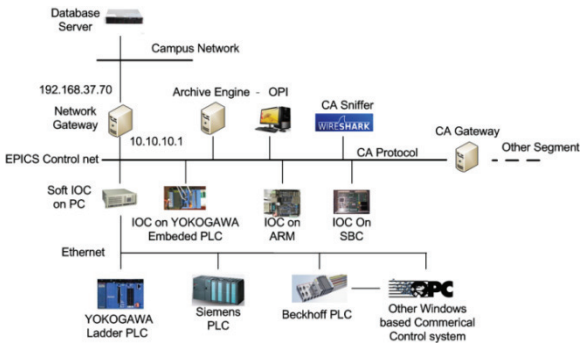


Figure 3: EPICS system architecture. In this figure, all components integrated into EPICS are shown.

Database and Network

MySQL is a choice for the experimental control system, according to its successful application in BESIII. The main advantage of MySQL is free. It can be easily to be developed in distributed environment to achieve database expansion, backup and accessing speed speedup.

Now, MySQL has been integrated into EPICS. Its functions and performance tested is in progress. A long term test and further study will be done next.

The control net spread in 80m x 80m experiment hall. Cables are routed in bridge along the wall. Fibers are used to avoided length limit and interferences Figure 4 shows the topology of the whole control net. Two redundant core switches in control room provide reliable net service. Access switches are placed closed to front device

Front Control

PLC from YOKOGAWA in Japan and Control system from Beckhoff in German are used in front control layer.

The two systems are also test in function test system, and have been integrated into EPICS system.

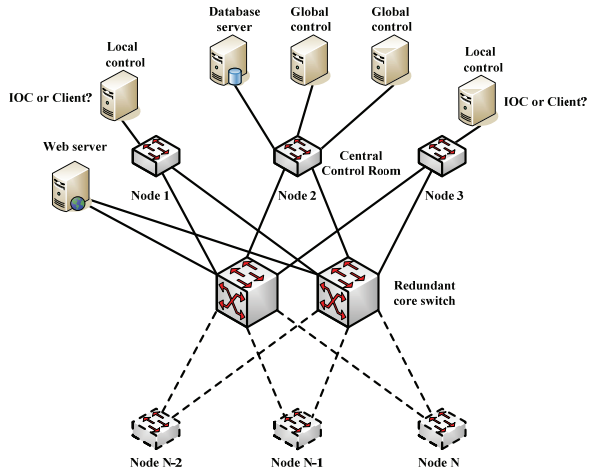


Figure 4: Global control net. The core switch is redundant to avoid shutdown of whole system. And links to core switch is also redundant. The failure of other switch only impact local sub-system.

TPS and PPS

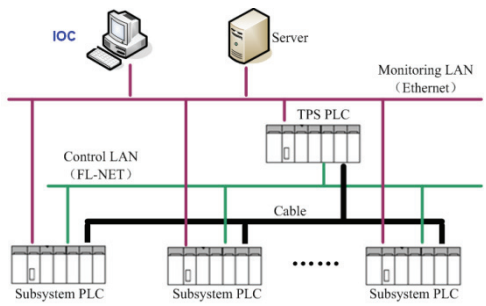


Figure 5: TPS architecture.

Design and technical study of TPS are nearly finished. Figure 5 shows the architecture of TPS. Key interlock signal is routed by hardware, and less important signals are communicated in FL_NET.

A function test system has been established with the heavy water control simulation involved in. A New developing system will setup in Beijing and Dongguan to fulfil the whole system development. Function test system will be upgraded to Beijing segment of the developing system. This function test system is shown in Figure 6.

EPICS software, network, database, archiving and others have all been tested in function test system.VNC, DNS and VPN are also tested which will help integrate the system in two locations into one system.

Function Test System

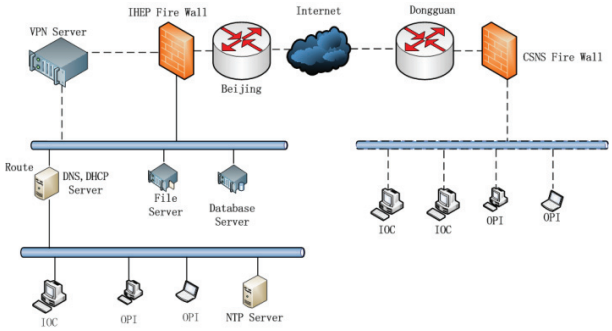


Figure 6: Structure of function test system now and future upgrade. Solid line means what we have already implemented. Dash line means the segment will be build when lab in Dongguan is ready.

Device Control

Some device level control is implemented by control group, which will be help the upper layer test and development. Figure 7 shows the sample platform in HIPD instrument. The platform is driven by 4 servo motors. Servo motors will be controlled by Beckhoff control system through 2 servo amplifier.

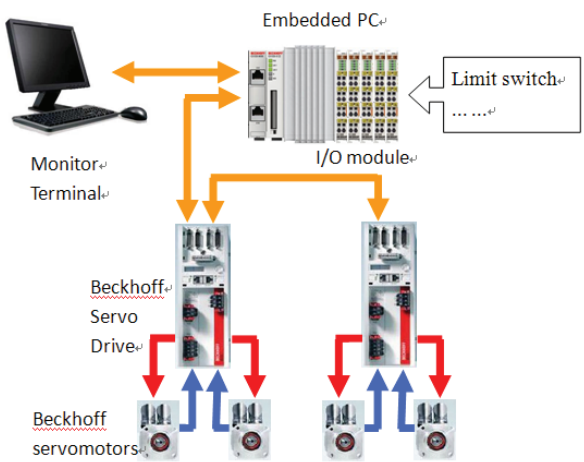


Figure 7: Architecture of sample of HIPD.

Sequence of Events Recording

For control system, sequence of event monitoring is useful for system self-check. On the other hand, sequence of events recording enables rapid root cause analysis after multiple events occurred. Sequence of events is therefore utilized as a diagnostic tool for trouble shooting and minimizing overall downtime. The system structure is shown in Figure 8.

XFC (eXtreme Fast Control Technology) modules from Beckhoff Company are used to build the sequence recording. A test result in Figure 9 shows that this system has a 1us time resolution with some dead time. Next, further study of sequence recording system to achieve higher time resolution to meet the requirements from T0 monitoring will be done.

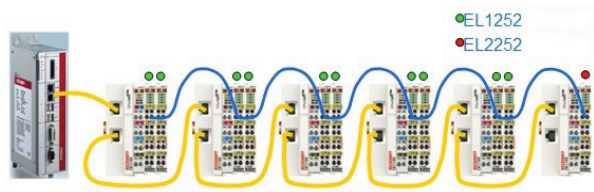


Figure 8: Time resolution test. EL1252 module sends a pulse and all EL2252 measure this pulse and takes a timestamp. The difference of these timestamp is the time resolution of the system.

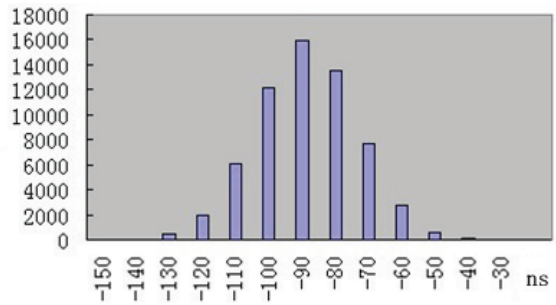


Figure 9: The time resolution of XFC module. X axis means the time difference measured on the same signal. Y axis delegates the sample numbers. The total sample numbers is more than 60,000. And this test also shows there is no big impaction of length of cable and sequence of modules.

NEXT TO DO

In the next two years, setup of the development system, a prototype system of entire CSNS experimental control system is key work. In this system, long-term stability and speed of MySQL; study of CSS and other tools in EPICS further; integration the device into EPICS; design of self-check and alarm system; design of fan-out system of T0 related signals; implementation of TPS and PPS will be done

SUMMARY

In the past, global control layer and some local layer design are finished. Some devices control is under developing now. The function test system is established. EPICS, front controllers and many other techniques have tested.

REFERENCES

[1] Wei Jie et al, China Spallation Neutron Source - an overview of application prospects , Chinese Phys. C, Volume 33, 2009.
[2] http://csns.ihep.ac.cn/english

THE LOCAL CONTROL SYSTEM OF AN UNDULATOR CELL FOR THE EUROPEAN XFEL

S. Karabekyan, R. Pannier, J. Pflüger, European XFEL GmbH, Hamburg, Germany
N. Burandt, J. Kuhn, Beckhoff Automation GmbH, Verl, Germany
A. Schöps, DESY, Hamburg, Germany

Abstract

The European XFEL project is a 4th generation light source. The first beam will be delivered in the beginning of 2015 and will produce spatially coherent ≤80fs short photon pulses with a peak brilliance of 10^{32}-10^{34} photons/s/mm^2/mrad^2/0.1% BW in the energy range from 0.26 to 29 keV at electron beam energies 10.5 GeV, 14 GeV or 17.5 GeV [1, 2]. Three undulator systems SASE 1, SASE 2 and SASE 3 are used to produce photon beams. Each undulator system consists of an array of undulator cells installed in a row along the electron beam. A single undulator cell itself consists of a planar undulator, a phase shifter, magnetic field correction coils and a quadrupole mover. The local control system of the undulator cell is based on industrial components produced by Beckhoff Automation GmbH and a PLC implemented in the TwinCAT system. Four servo motors control the gap between the girders on each undulator with micrometer accuracy. One stepper motor is used for phase shifter control, and two other stepper motors control the position of the quadrupole magnet. The current of magnetic field correction coils as well as the gap of the phase shifter are adjustable as a function of the undulator gap. The high level of synchronization (<<1μs) for the complete undulator system (for instance SASE 2 with 35 undulator cells in total) can be achieved due to implementation of a fast EtherCAT fieldbus system in the local control.

SYSTEM OVERVIEW

At the project start-up stage three undulator systems SASE1, SASE2 and SASE3 will be used to produce photon beams (see Figure 1). The electron bunch train is distributed into two branches by a flattop kicker magnet into SASE1 and SASE2 beam lines, where hard X-ray beams are generated. After passing through SASE1 the electron bunches are used a second time by passing through the SASE3 undulator system to create additional soft X-ray beam.

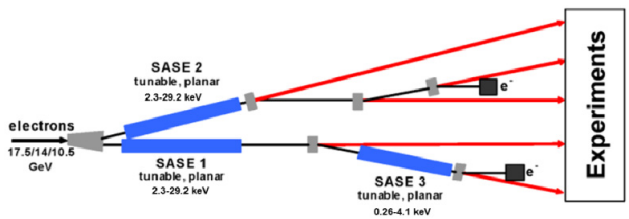


Figure 1: Schematic layout of the electron and photon beam distribution.

The parameters of the undulator systems relevant to the control system are shown in the Table 1 [1-3].

Requirements

The tolerance requirements for the undulator systems relevant to the control system are following [1-6]:

Table 1: Parameters of the Undulator Systems Relevant to the Control System

Photon Beamline	Electron energy GeV	Photon energy keV	Wavelength Å	Gap mm	Magnetic period mm	Quantity of Undulators
SASE 1	10.5	2.3 – 14.9	5.4 – 0.83	10 – 24		
&	14	4.1 – 18.7	3.0 – 0.66	10 – 20	40	35
SASE 2	17.5	6.4 – 29.2	1.9 – 0.43	10 – 20		
	10.5	0.26 – 2.2	47.7 – 5.6	10 – 28		
SASE 3	14	0.47 – 2.6	26.6 – 4.8	10 – 24	68	21
	17.5	0.73 – 4.1	16.9 – 3.0	10 – 24		
						Total: 91

- Undulator gap control accuracy ±1 μm.
- Quadrupole mover positioning repeatability ±1 μm.
- Phase shifter gap control accuracy ± 10 μm.
- Max. Steering Power for Air Coil Correctors ±0.6 Tmm.
- Accuracy of the temperature measurement of magnet structures ± 0.03 K.

The functional requirements to the local undulator cell control are:

- Gap control with low following error (<±10μm).

- Local temperature measurement and appropriate undulator gap correction.
- Undulator gap dependent air coil correction.
- Undulator gap dependent phase shifter control.
- Motion control for quadrupole movers.
- 3 way mixing valve control for the beam pipe temperature stability.
- Safe operation, damage prevention, proper and precise movement limitation, failure detection.

An undulator cell consists of a 5m long undulator segment and a 1.1m long intersection segment (see Figure 2). Four servo motors are used on each undulator to control the gap between girders with micrometer accuracy. One stepper motor is used for phase shifter control, and two other stepper motors control the position of the quadrupole magnet. The current of magnetic field correction coils as well as the gap of the phase shifter are adjustable as a function of the undulator gap.

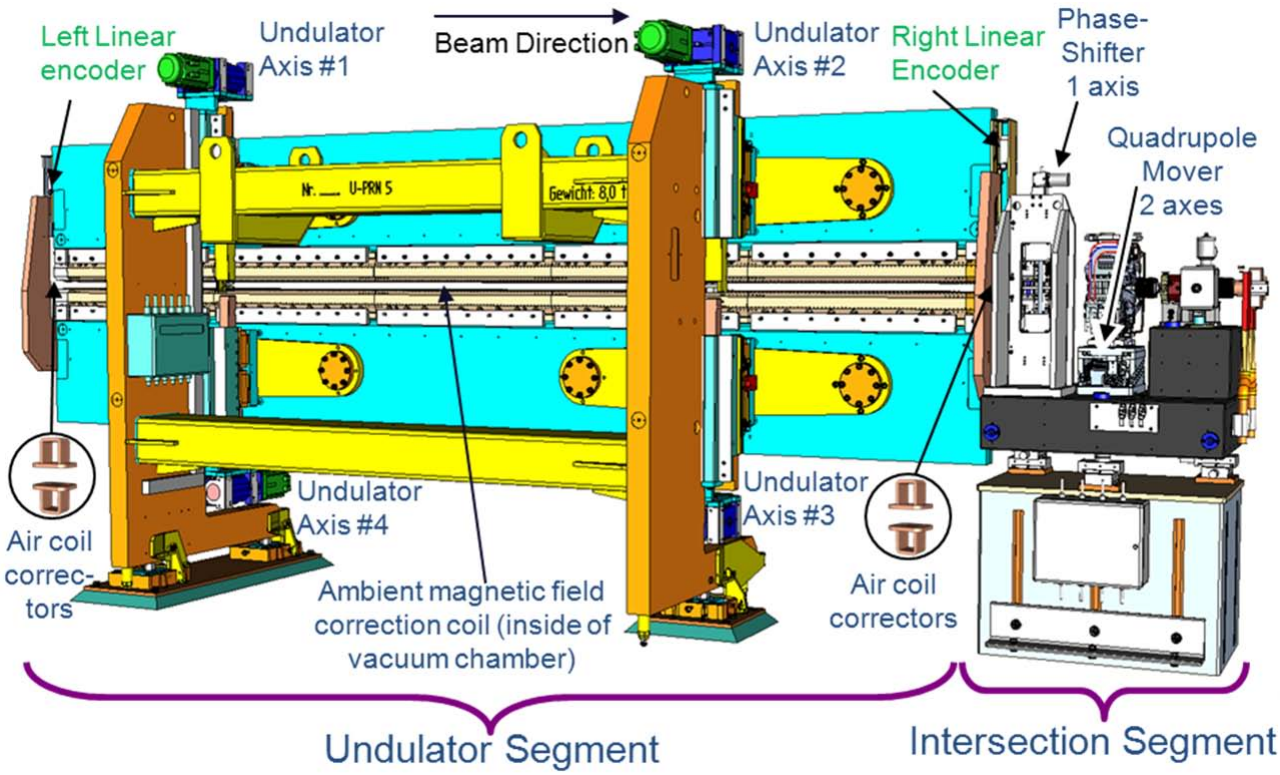


Figure 2: Undulator cell. Undulator and intersection segments in array.

UNDULATOR CELL CONTROL

Control of the Undulator Gap

Each of the four undulator motors is equipped with a rotary multiturn absolute encoder, flanged directly on the axis. In addition to those four encoders each undulator is equipped with two absolute linear encoders, which are installed on both ends of the undulator girders. These linear encoders directly measure the right and left gap between the girders. The undulator can be operated either using rotary encoders or linear encoders as a feedback for the servo drivers. At small gaps the strong magnetic forces cause a deformation of the undulator support frame and thus cause deviations between the linear and the rotary encoder readings. To compensate the influence of these deformations the gap is measured with high precision external gauges during commissioning. The results of these measurements are used to generate curves (see Figure 3) which are implemented as feed forward corrections farther in the PLC program.

If the rotary encoders are used for the gap control, then these correction curves are applied to all four axes.

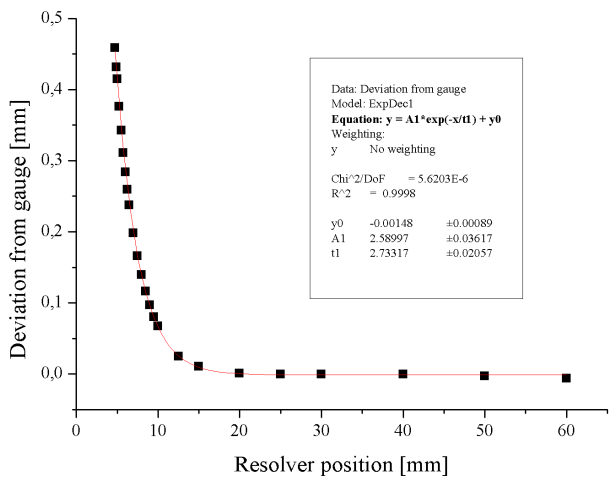


Figure 3: Evaluation of the correction curves for one axis.

In case of using the linear encoders as feedback, the lower two axes are using the corrected value of the rotary encoders as a feedback, while the position of the upper two axes is controlled according to the readings of the linear encoders.

Temperature Drift Compensation

The NdFeB permanent magnet material, which is used to create the magnetic field in the undulator, has a temperature coefficient for its remanent field. These temperature coefficient for the relative magnetic fields $(\Delta B/B)/\Delta T$ in the air gap of the NdFeB dipole magnets is $\sim -1.1\cdot10^{-3}$ K^{-1}. To compensate for magnetic field changes due to temperature variations the gap correction method is used [4]. The required gap correction is calculated in the PLC program. The correction is done according to equation 1:

$$\Delta g_{Loc} = \frac{\lambda_U \eta}{b + 2cg / \lambda_U} \Delta T_{Loc} \qquad (1)$$

where $\Delta T_{\mathrm{Loc}} = T_{\mathrm{Nom}} - T_{\mathrm{Loc}}$, T_{Loc}, is the local temperature, T_{Nom} is the nominal operating temperature of the undulator system, λ_U is the undulator period length, g the undulator gap, η is the reversible temperature coefficient of NdFeB ($-1.1\cdot10^{-3}$ K^{-1}), b and c are empirical constants describing the gap dependence of the peak field.

At 10mm gap, for instance, the temperature dependence for SASE 1 and SASE 2 is ~9.17µm/K, for SASE 3 this dependence is ~15.7µm/K [5].

To provide accurate temperature data three PT100-3 sensors are mounted inside the magnetic structures, one in the middle of the upper structure and two on both edges of the lower structure. The temperature is measured by Almemo 8590-9 Delta-sigma, 24-bit A/D converter.

Control of the Temperature of Vacuum Chamber

In order to avoid bending of the magnet girders by temperature gradients the temperature of the vacuum chamber should not differ from that of the girders. Thermal stabilization is achieved through the cooling water by appropriate mixing of warm (27°C) and cold (18°C) water with 3-way valve. The actual water temperature of the 3-way valve outflow is measured by PT100-3 sensor, which is connected to the same Almemo 8590-9 temperature measuring device, and provides feedback to the PLC program. In the PLC program the water temperature and the temperature of magnetic structure are compared and 3-way valve is controlled to eliminate any deviation.

Magnetic Field Corrections by Means of Air Coils

On each undulator segment two horizontal and vertical air coil correctors are used.

- to compensate residual gap dependent steering errors of the undulator (~ ± 0.1 Tmm),
- to compensate residual gap dependent steering errors of the phase shifter (~ ± 0.05 Tmm),
- for beam ballistic steering of ± 0.45 Tmm

The maximal steering power in horizontal and vertical direction is therefore ±0.6 Tmm.

During operation the air coil correctors are controlled using look up tables. These look-up tables contain the steering strengths as a function of the undulator gap, which are required to compensate 1st and 2nd field integral errors. They need to be derived from magnetic measurements. The required correction currents are calculated from the conversion constants which are in the range 0.4 to 0.67 Tmm/A.

An ambient magnetic field correction coil consisting of just two parallel wires is fitted inside two bores of the vacuum chamber. It's called Two Wire Corrector (TWC). It can be used for compensation of an ambient magnetic field of up to 150µT.

The current for each air coil and the TWC is regulated by means of constant current power supplies controlled via analog output terminals. Direction of the magnetic field is changed by means of polarity reversal relay, which is changing the current direction supplied to the air coil.

Phase Shifter Control

For gap adjustable undulator systems phase shifters are need to adjust the phase between electrons and photons field. The phase shifter for European XFEL is based on permanent magnet technology. The magnet structure consists from four magnetic arrays, two in the top and two in the bottom. The phase is adjusted by changing the gap between upper and lower magnetic arrays. Figure 4 shows the dependence of the phase shifter gap value on the undulator gap value for SASE 2 and SASE 3 at different harmonic numbers [6]. The basic control requirement is that the phase shifter gap has to follow the undulator gap.

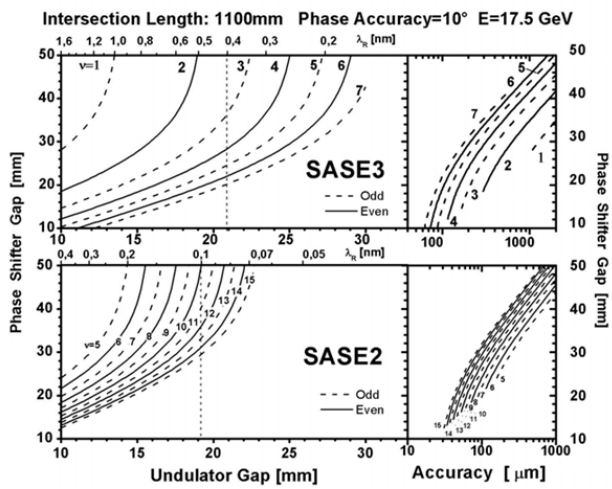


Figure 4: Tuning curves for the phase shifter. The right-hand side graphs show the required gap precision of the phase shifter for to control the phase within ±10°.

The phase shifter is controlled by means of a look up table, which is evaluated from the magnetic tuning curves. Both motion controls, undulator and phase shifter, are synchronized with a following error of ≤10µm.

Motion control for the phase shifter consists of a five-phase stepper motor, a self-locking gearbox with a ratio of i=50, a spindle with right- and left-handed thread with 5 mm pitch, and an incremental linear encoder for position feedback (see Figure 5).

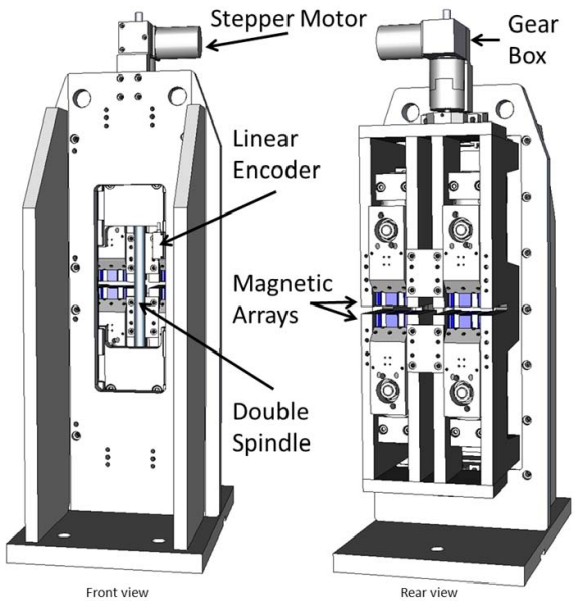


Figure 5: Motion control components of the phase shifter.

Quadrupole Mover Control

The control of the quadrupole magnet movers that are situated between undulator segments is a part of the local undulator control system as well. Information about the quadrupole magnet corrections or the set values in horizontal and vertical directions is obtained from the beam positioning system. This information is received by undulator local control from the accelerator control via the global undulator control system. The requirements to the quadrupole mover control are the following:

- Movement range in horizontal and vertical directions: ±1.5 mm.
- Positioning repeatability in both directions: ±1µm.
- Maximal movement speed in both directions: 1mm/sec.
- Maximum load: 75kg

The quadrupole mover control consists of two actuators for horizontal and vertical movement, driven by five-phase stepper motors and two LVDT sensors as feedback for each motor.

Implementation

The local control system of undulator cell is completely implemented in Beckhoff's PLC and TwinCAT system manager.

The graphical user interface (GUI) consists of following windows:

- Main control window.
- Intersection control.
- Alarm display.
- Axes status.
- System information.

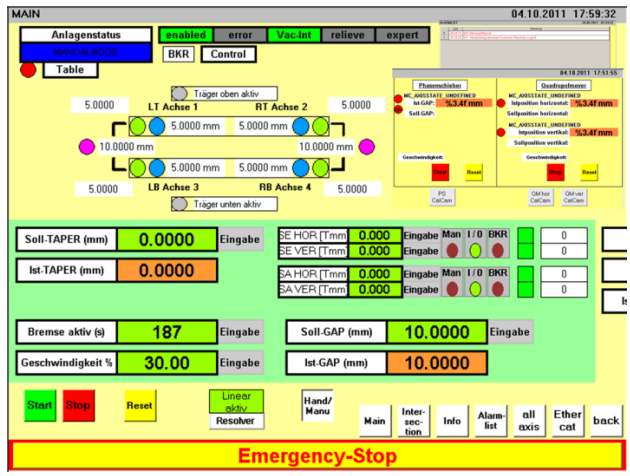


Figure 6: GUI widows of the local control for the undulator cell.

The local control system provides all possibilities for control, monitor and error tracing of each undulator cell. It also provides the interfaces to integrate the local control system into the global undulator control system.

REFERENCES

[1] M. Altarelli et al., Eds., "The European X-ray Free-Electron Laser: Technical Design Report", DESY 2006-097, July 2007

[2] T. Tschentscher, "Layout of the X-Ray Systems at the European XFEL, Photon Beam Systems Meeting", 29 March 2011

[3] Yuhui Li, Bart Faatz, and Joachim Pflueger, "Undulator system tolerance analysis for the European x-ray free-electron laser", Phys. Rev. ST AB 11, 100701 (2008).

[4] Hedqvist A., Hellberg F., Danared H, Karabekyan S. and Pflüger J., "Precision temperature measurement system for the XFEL undulator systems", Conceptual design report.

[5] J. Pflüger, R. Pannier, "PETRA III Air Coil Correctors adopted for XFEL", XFEL Report WP71/2009/01.

[6] H.H. Lu, Y.Li, J.Pflueger, "The permanent magnet phase shifter for the European X-ray Free electron laser", Nucl. Instr. and Meth. A 605 (2009) 399–408.

PHASE II AND III THE NEXT GENERATION OF CLS BEAMLINE CONTROL AND DATA ACQUISITION SYSTEMS *

E. Matias#, D. Beauregard, R. Berg, G. Black, M.J. Boots, W. Dolton, D. Hunter, R. Igarashi, D. Liu. D. Maxwell, C. D. Miller, T. Wilson, G. Wright
Canadian Light Source, University of Saskatchewan, Saskatoon, Canada

Abstract

The Canadian Light Source is nearing the completion of its suite of Phase II Beamlines and in detailed design of its Phase III Beamlines. The paper presents an overview of the overall approach adopted by CLS in the development of beamline control and data acquisition systems. Building on the experience of our first phase of beamlines the CLS has continued to make extensive use of EPICS with EDM and QT based user interfaces. Increasing interpretive languages such as Python are finding a place in the beamline control systems. Web based environment such as ScienceStudio have also found a prominent place in the control system architecture as we move to tighter integration between data acquisition, visualization and data analysis.

PHASE II AND III BEAMLINES

The Canadian Light Source (CLS) is a 2.9 GeV Synchrotron facility[1] that contains six phase I beamlines in operation, an additional seven phase II beamlines in various stages of development (as listed in Table 1) and an additional seven Phase 3 beamlines under construction. Table 2 provides a comparison of underlying technology used across all of the CLS beamlines.

Table 1: Phase II Beamlines

Name	Application	State
BMIT-BM	Biomedical Imaging	Operational
BMIT-ID	Biomedical Imaging	Commissioning
CMCF2	Protein Crystallography	Operational
REIXS	Emission Spectroscopy Resonant X-ray Scattering	Operational Commissioning
SXRMB	XAFS Micro-Probe	Operational Commissioning
SyLMAND	Lithography	Commissioning
VESPERS	Spectroscopy & Micro-Probe	Operational

To minimise project risk and ensure the cost effective delivery of the beamline control systems the underlying architecture developed for the first phase of beamlines[2] was to a large extent applied again to the Phase II and Phase III development.

EPICS was again used as the underlying platform of the development of CLS new round of beamlines. This permitted us to leverage significant earlier development.

COMMON ELEMENTS

Operating Systems

Linux continued to be the primary operating system platform used at CLS. This was augmented with a more limited number of MS-Windows based data acquisition computers required to support specific commercial off-the shelf (COTS) software.

Though CLS has extensive experience with the use of RTEMS, we have continued to find the need for hard-real real-time applications limited to certain accelerator applications. We continue to evaluate the potential application of RTEMS to our beamlines.

Hardware Platform

We find the use of the Moxa MIPS based controllers (Model UC-7408-LX and DA-662-16-LX) compelling[3]. Over the past four years CLS have deployed 120 of Moxa computers in the facility and has yet to experience any significant hardware failure. The Moxas are used as full fledged EPICS IOCs primarily interfacing Ethernet, USB and Serial devices into the control system.

Modicon Momentum PLCs continue to be used primarily for machine protection. These are interfaced into the CLS control system through Modbus over TCP/IP.

VME64x is primarily used for motion control and data acquisition applications. The Prodex MaxV is used with a combination of drivers [4]. Data acquisition is primarily accomplished with the use of SIS3820 scalar boards.

Network Setup

Each beamline has its own VLAN, with all of the beamline specific IOC and OPI computers located on the VLAN. Any computers that a user brings in to control a user-furnished end-station are also added to this VLAN.

EPICS IOC applications, home directories, applications and users data storage areas are accessed by NFS from a central SAN.

For added security, Ethernet devices are normally connected directly on a secondary network port on a Moxa and are not directly accessible on the VLAN. Beamline specific PLCs are also normally placed on the

*Research described in this paper was performed at the Canadian Light Source, which is supported by the National Sciences and Engineering Research Council of Canada, the National Research Council of Canada, the Canadian Institute for Health Research, the Province of Saskatchewan, Western Economic Diversification Canada, and the University of Saskatchewan.
elder.matias@lightsource.ca

accelerator VLAN and accessed through EPICS Gateway software.

Table 2: Comparison between All CLS Phase I,II, III Beamlines

Beamline Name	Machine Protection	Motion Control	Beamline Display	Experiment Control	Experiment User Interface
Brockhouse	Modicon Momentum	Prodex MaxV Newport XPS	EDM	CLS Scan Lib	ScienceStudio Others TBD
BioXAFS	Modicon Momentum	Prodex MaxV Newport XPS	EDM	CLS Scan Lib	CLS-IDA TBD
BMIT-BM	Modicon Momentum	Prodex MaxV Labview	EDM	BMIT Specific	3rd Party Vendor
BMIT-ID	Modicon Momentum	Prodex MaxV Bosch-Rexroth Labview	EDM	BMIT Specific	3rd Party Vendor
CMCF1	Modicon Momentum	Prodex OMS58 Prodex MaxV	EDM	Python	MxDC MxLive
CMCF2	Modicon Momentum	Prodex MaxV	EDM	Python	MxDC MxLive
Far-IR	Modicon Momentum	Prodex MaxV	EDM	3rd Party Vendor	3rd Party Vendor
HXMA	Modicon Momentum	Prodex MaxV Newport XPS	EDM	CLS Scan Lib	CLS -IDA Spec
IDEAS	Modcon Momentum	Prodex MaxV	EDM	CLS Scan Lib	CLS –IDA
Mid-IR	Modicon Momentum	Prodex MaxV	EDM	3rd Party Vendor	3rd Party Vendor
QMSC	Modicon Momentum	Prodex MaxV	EDM	CLS Scan Lib	3rd Party Vendor
SGM	Modicon Momentum	Prodex OMS58	EDM	CLS Scan Lib	CLS-IDA
PGM	Modicon Momentum	Prodex OMS58	EDM	CLS Scan Lib	CLS-IDA
REIXS	Modicon Momentum	Prodex MaxV	EDM	CLS Scan Lib	Aquaman Spec
SM	Modicon Momentum	Prodex OMS58 Newport XPS	EDM	Beamline Specific	3rd Party Vendor
SXRMB	Modicon Momentum	Prodex MaxV Newport XPS	EDM	CLS Scan Lib	CLS –IDA
SyLMAND	Modicon Momentum	Proxed MaxV	EDM	3rd Party Vendor	3rd Party Vendor
VESPERS	Modicon Momentum	Prodex MaxV	EDM	CLS Scan Lib	CLS –IDA Aquaman ScienceStudio

Centrify is used to provide unified cross-platform user authentication for access to networked resources such as password authentication and networked file access. This ensures account information is the same on all workstations in the facility, and changes to account information are available immediately.

Virtual Machines

CLS now has two years of experience in the use of VMware ESX based virtual machines in the control system[5]. A common ESX cluster is used for the control system across the accelerator and all of the beamlines. softIOC applications and data analysis applications that do not require specific hardware access are deployed on virtual machines.

EPICS

EPICS continues to be used as the common software platform for integrations of the various sub-systems. The EPICS Gateway provides access to storage ring devices, such as insertion devices.

From the synApps toolkit [6], developed at APS, CLS makes use of asyn and areaDetector libraries. These two libraries are used to interface most of the serial devices found and detectors found on the beamlines.

CLS Extensions to EPICS

The base EPICS software is supplemented by the use of locally developed scanning libraries and experiment specific higher level beamline applications [7].

CLS makes extensive use of the Qt toolkit in building custom extensions. The CLS-IDA and Aquaman are two example of such applications.

Increasingly Python has started to play a more significant role in the development of user applications. The CMCF 1&2 beamlines operator interface software is based on Python and work is underway on the development of an EDM compatible display tool based on Python.

Relational Databases

Predomently mySQL is used for relational databases. MxLIVE, MxDC, and ScienceStudio all make use of mySQL.

BEAMLINE SPECIFIC ELEMENTS

BMIT

The BMIT beamlines use standard CLS motor controls (EPICS, VME, MaxV) for operation of the beamline equipment installed in the optics hutches (including monochromators, filters, slits, shutters etc.). High-level software for device control and beamline monitoring has also been developed using EPICS and EDM screens. Wherever possible, high-level beamline software used by researchers provides an abstraction layer to hide the details of operation of physical beamline components from the user, allowing them to instead specify and achieve their end goal.

Standard CLS control software is supplemented on BMIT with the use of additional software products. LabView programs are used to operate smaller positioning devices within the experimental hutches. Use of this equipment varies widely for different experiment types and sample sizes, and LabView provides beamline staff with the ability to quickly adapt to various experimental configurations. The three large positioning systems on BMIT use Bosch servo motors with low level control functionality provided by Bosch Rexroth IndraMotion MLD drive-integrated PLC programs developed using IndraWorks software. Data acquisition on BMIT is primarily via vendor software particular to the detectors in use on the beamline.

CMCF2

The phase 1 and phase 2 Protein Crystallography beam lines have two major components, the local data collection software, and a web interface. The data collection software, called MxDC [8], was written in house by Michel Fodje, in python using GTK+ (Glib) (reference) as the main application framework with Twisted and Avahi. Access to the beam line for control is accomplished through Epics Channel Access for Python (reference). MxDC was deployed onto the beam line for production use in 2007 and has seen extensive feature additions through constant use and testing as well as user feedback. Using python as the development language has made it easier to interface the data collected locally by users at the beam line with the web based interface MxLIVE.

MxLIVE allows users to describe their samples, submit their experiments, approve strategies, and review results over the web. Developed on the django web framework (Python), MxLIVE is independent of the data collection software, as all communication is through a relational database (MySQL, json). Users may view results as soon as they are collected, and download individual images as well as full datasets directly from MxLIVE. MxLIVE is very popular with both “mail-in” users and users who are coming to the CLS to collect their own data. It was first made available to users in May 2011.

Some sophisticated users who prefer not to travel to the CLS are also enabled to use the data collection software remotely, via a NoMachine interface.

REIXS

REIXS beamline control was based on CLS standard EPICS controls including MAXv motor control and an EPICS based energy application originally developed for the Spectro Microscopy (SM) beamline. The energy application allows simultaneous setting of the monochromator mirror and grating and the gap and girders of one or two elliptically polarizing undulators (epus). Energy scans can be accomplished with one epu or two epus in conjunction with a chopper. Spec is used for RSXS data acquisition and “Aquaman”[9] software developed in a collaboration between the University of Saskatchewan and CLS for XAS and XES data acquisition.

REXIS also contains a limited amounted of SPEC to meet some user specific requirements.

SXRMB

The success of the controls at SXRMB is a reflection of the greater success of software standards at the CLS. Commonality of drivers allows quick reuse of software at both the driver and GUI levels. The initial XAFS commissioning experiments used the HXMA beamline’s C++/Qt IDA application, as well as a number of EDM support screens. This was followed by a customized version of the XAFS software (C++/Qt) used at the SGM beamline, which gives a more integrated data acquisition environment. The microprobe endstation is near

completion, and will be able to use the multi-dimensional scanning and acquisition software (C++/Qt/Root) in use at the VESPERS and HXMA beamlines.

SyLMAND

SyLMAND is also an EPICS based beamline. The Endstation is a pre-pacakged scanner with vendor furnished software running on an MS-Windows platform. To facilitate integration with the beamline, the vendor was provided with EPICS interface libraries for Visual Studio C++.

VESPERS

VESPERS makes use of two different high-level experiment management frameworks: (1) Aquaman and (2) ScienceStudio, Both system are the same underlying EPICS IOC applications and frameworks for device control permitting rapid switching between the two systems.

Acquaman [9] is an experiment-centred framework that encapsulates low level control into coherent objects and exposes, to the end user, only what is necessary to run scientific experiments on the beamline. This enables users with varying levels of synchrotron experience to take data effectively without the need for extensive software training. The primary benefits are three-fold. First, the process of switching between experiment settings is presented in obvious ways without the need of exposing low level hardware. As well, where only the pertinent pieces of the beamline are exposed to users, the chances of a user accidentally accessing part of the beamline that they should not are minimized. The Acquaman framework has been applied on VESPERS to create a tailored user interface capable of seamlessly supporting XRF and XAS. Lastly, all data that is collected using the Acquaman framework is automatically saved into a database to ensure that no data that is collected can be inadvertently lost and can be exported at the leisure of the user.

Currently, the user interface handles all general features of the beamline, such as photon shutters, slits, sample position, and positions for all detectors. The Acquaman framework allows for easy control of the beamilne while enforcing logic that prevents users from entering situations that may damage the beamline (ie: colliding detectors together). The user interface also configures the silicon drift detectors (SDD) for setting up X-Ray Fluorescence (XRF) 2D maps and allows easy access to configuring regions of interest which provides tools to quickly assess the elemental composition of the samples.

Finally, encapsulation of all necessary controls to perform X-ray Absorption Spectroscopy (XAS) while still hiding low level control that will enable users to concentrate on their experiment, rather than compromising their data with low level software.

Science Studio [10] is a web portal, and framework, that provides scientists with a platform to collaborate in distributed teams on research projects, and to remotely access scientific resources. Like Aquaman ScienceStudio only exposes the aspects of the beamline that are pertinent to the user experience. The core component of Science Studio is a web-based experiment management system which allows users to organize and share their sample information, experimental data and analysis results. The VESPERS beamline has been integrated into Science Studio to provide remote control and data acquisition capabilities. ScienceStudio has been extended to provide on-line realtime analysis of both XRF and XRD data, while using high performance computing to process the XRD data.

REFERENCES

[1] L. Dallin, I. Blomqvist, M. de Jong, D. Lowe, and M. Silzer, “The Canadian Light Source,” PAC’03, Portland, May 2003, p. 220-223 (2003); http://www.JACoW.org.

[2] E. Matias, et al, “The Canadian Light Source Control System: Lessons Learned From Building a Synchrotron and Beamlines Control System,” ICALEPS’05, Geneva, Oct 2005; http://www.JACoW.org.

[3] E. Matias, et al., “The CLS – A Case Study in the Use of Single Board Computers, and Industrial PC Equipment for Synchrotron Control,” PCaPAC’08, Ljubljana, Oct. 2008, WEZ03, p. 157-158 (2008); http://www.JACoW.org.

[4] D. Bertwistle, E. Matias, M. McKibben, “Experience with Motion Control Systems at The Canadian Light Source,” ICALEPS’09, Kobe, Oct. 2009, WEP05, p. 501-503 (2009); http://www.JACoW.org.

[5] G. Wright, C. Angel, C. Finlay, E. Matias, “Migrating Control Servers and Applications to Virtual Machines,” ICALEPS’09, Kobe, Oct. 2009, WOD06, p. 43-45 (2009); http://www.JACoW.org.

[6] T. Mooney. “synApps: EPICS Application Software for Synchrotron Beamlines and Laboratories,” PCaPAC’10, Saskatoon, Oct. 2010, THCOM02, p. 106-108 (2010); http://www.JACoW.org.

[7] G. Wright, R. Igarashi, “EPICS Data Acquisition Software at the CLS,” PCaPAC’10, Saskatoon, Oct. 2010, THPL009, p. 142-143 (2010); http://www.JACoW.org.

[8] M. N. Fodje, R. Berg, G. Black, P. Grochulski, K. Janzen, “Automation of the Macromolecular Crystallography Beamlines at the Canadian Light Source,” PCaPAC’10, Saskatoon, Oct. 2010, THPL005, p. 130-132 (2010); http://www.JACoW.org.

[9] D. K. Chevrier, M. Boots, “Experiment Based User Software,” PCaPAC’10, Saskatoon, Oct. 2010, FRCOAA04, p. 211-213 (2010); http://www.JACoW.org.

[10] http://www.sciencestudioproject.com

DEVELOPMENT OF DISTRIBUTED DATA ACQUISITION AND CONTROL SYSTEM FOR RADIOACTIVE ION BEAM FACILITY AT VARIABLE ENERGY CYCLOTRON CENTRE, KOLKATA

K. Datta#, C. Datta, D. P. Dutta, D. Sarkar, H. K. Pandey, T. K. Mandi, VECC, Kolkata, India
A. Balasubramanian, K. Mourougayane, R. Anitha, SAMEER, Chennai, India

Abstract

To facilitate frontline nuclear physics research, an ISOL (Isotope Separator On Line) type Radioactive Ion Beam (RIB) facility is being constructed at Variable Energy Cyclotron Centre (VECC), Kolkata. The RIB facility at VECC consists of various subsystems like Electron Cyclotron Resonance (ECR) Ion source, analysing magnet, Radio Frequency Quadrupole Linear Accelerator (RFQ linac), Rebunchers, Inter-digital H-mode Linear Accelerators (IH linacs) etc which are required to produce and accelerate radioactive ions for different experiments. To facilitate the smooth operation of this RIB facility by way of monitoring, supervision and control of all the important parameters associated with its various sub-systems, a "Distributed Data Acquisition and Control System (DDACS)" is being developed at this Centre. This paper briefly describes the design philosophy, basic architecture and various functional components of the DDACS.

INTRODUCTION TO RIB FACILITY

The RIB facility (Fig. 1) at VECC is an ISOL type RIB facility. Radioactive isotopes are first produced in a target from nuclear reaction with the primary beam coming from K-130 Cyclotron operational at this Centre. Radioactive atoms diffusing from the target are ionized initially in an integrated 1^+ ion-source and then in a 6.4 GHz Electron Cyclotron Resonance Ion source (ECRIS). After mass separation, the low energy RIB is accelerated from 1.75 KeV/u to about 100 KeV/u in Radio Frequency Quadrupole Linear Accelerator (RFQ linac). Three Inter-digital H-mode Linear Accelerator (IH linacs) modules raise the energy up to around 415 keV/u. Subsequent linac modules are to be used to achieve the final beam energy of 1.3 MeV/u [1].

DESIGN PHILOSOPHY OF DDACS

The DDACS for RIB facility, like any other control systems of large accelerators, is not only a vital means of monitoring the status of the machine at any point of time during operation, but also a tool for achieving desired beam. A detailed requirement analysis was done for the design of the control system. Following are some of the guidelines which are followed in global stages and have also been appropriately adopted in the design of DDACS.

- The control system should have multi-layered architecture. Entire task of data acquisition and control would be shared by different control modules distributed in all layers [2].
- The control system must be modular, incrementally upgradeable and extendable to fulfil the future requirements as machine grows in scale [3].
- The modules of the control system must be individually usable in case of testing, installation and commissioning of a subsystem.
- The control system must have provision to accommodate various kinds of heterogeneous equipments manufactured by different companies and hence characterised by different operational requirements.
- Variation in design of individual control modules should be minimized. Homogeneous design of control modules must be of paramount importance for the ease of maintenance and replacement.
- Use of state-of-the-art technology.
- The control system should be expandable in terms of addition of similar modules for similar subsystems.

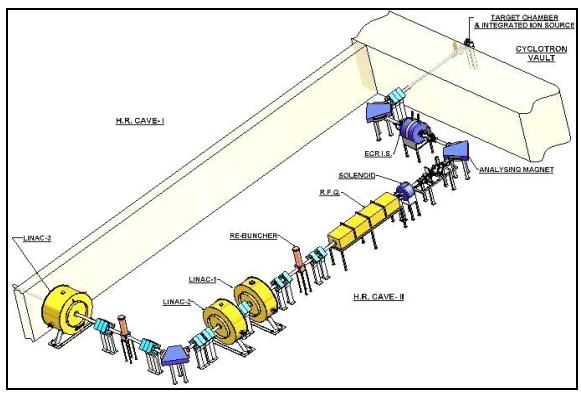


Figure 1: Layout of RIB facility (up to LINAC-3).

ARCHITECTURE OF DDACS

Based on the above design philosophy, RIB control system is being developed using 3-layer architecture: a) Equipment Interface Layer b) Supervisory Control Layer and c) Operator Interface Layer. The control system is composed of a number of functional modules distributed over these layers which are connected through suitable communication network and protocol.

The Equipment interface layer consists of different Equipment Interface Modules (EIMs) which are directly connected to the accelerator equipments through various interfaces. Each EIM is a microcomputer based intelligent unit with front-end analog/ digital electronics. Each of the EIMs is separately connected to the supervisory computer through fibre optic link in a star topology.

kaushikdatta@vecc.gov.in

The Supervisory layer has been realised using Embedded Controllers (EC) designed around Single-Board-Computer (SBC) with Embedded XP operating system. This controller, interfaced with EIMs through fibre optic cables, performs supervisory task of continuously sending command and acquiring data from lower level EIMs and reporting to the operator interface layer as and when requested.

Operator interface layer consists of operator console formed with another embedded controller and high performance PCs/ Workstations for controlling and monitoring machine parameters. The data acquired and analysed at the supervisory layer can be displayed at the operator console and operators can control the whole facility from the user friendly graphical interfaces.

A schematic diagram of the architecture is shown in Figure 2.

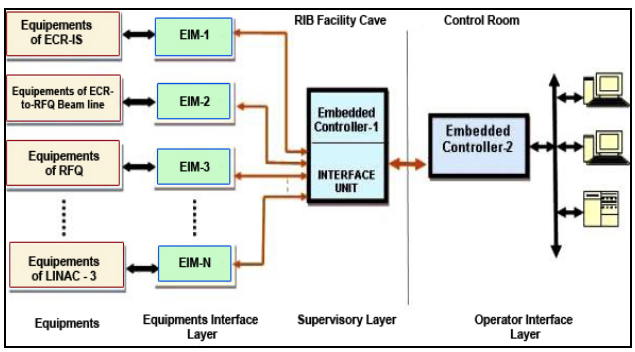


Figure 2: Schematic diagram of DDACS.

DESCRIPTION OF DDACS

For the development of data acquisition and control system, entire RIB facility has been viewed as a composition of following subsystems and their interconnected beam lines (as per present status of RIB facility).

a) Electron Cyclotron Resonance Ion source (ECRIS)
b) ECRIS to RFQ Beam Line
c) Radio Frequency Quadrupole (RFQ)
d) RFQ to Rebuncher Beam Line
e) Rebuncher
f) Rebuncher to Linac-1 Beam Line
g) Linac-1
h) Linac -1 to Linac-2 Beam Line
i) Linac -2
j) Linac -2 to Linac -3 Beam Line
k) Linac -3

These subsystems and their interconnected beam lines are associated with various kinds of equipments like High Current Magnet Power Supplies, High Voltage Power Supplies, Klystron High Power Amplifier (KHPA), RF Transmitters, Vacuum Pump Controllers, Vacuum Gauges, Gate Valves, Faraday Cups etc. DDACS is intended to monitor and control important parameters of these equipments and thus help smooth operation of machine.

The three layers of DDACS are detailed in the following sections.

Equipment Interface Layer (Layer-1)

The Equipment Interface Layer is the lowest level of control system which is directly interconnected to the accelerator equipments. It consists of multiple microcomputer based modules named as "Equipment Interface Module" (EIM) (Fig. 3) which form the basic foundation of this data acquisition and control system. The multiple equipments associated with each subsystem are physically connected to their respective EIM module through a variety of interfaces which actually governs the design of the EIM. Wide range of computer/remote interfaces available with the equipments from different manufacturers has been extensively studied and EIMs are finally designed to support multiple analog/ digital input/ output channels and RS-232/ RS-485 ports suitable for connecting all the equipments. Following a generalised hardware architecture, EIMs are indigenously developed around 32-bit ARM controller with analog/ digital front-end electronics (ADCs, DACs, Opto-isolators, Multiplexers etc.) and RS232/ RS-485 level translator. It also consists of a touch-screen display (Fig. 4) using which, equipments, connected to it, can be locally controlled and monitored. This feature immensely helps in tuning of the control system during installation and also in testing and commissioning of many subsystems.

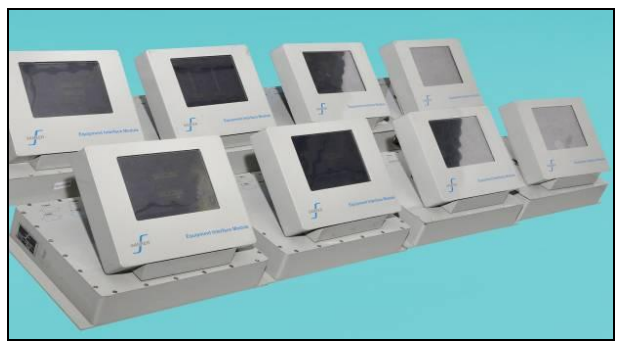


Figure 3: Snapshot of Equipment Interface Modules (EIMs).

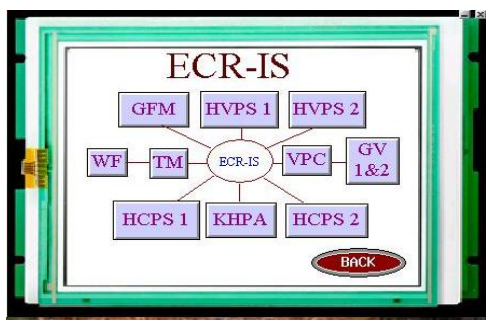


Figure 4: Snapshot of a typical touch-screen based local display panel of EIMs.

Some of the equipments such as RF transmitters, Faraday cups and Slit controller do not have compatible

remote interface to connect them to EIM. Hence, suitable interface units are introduced to enable them to get connected to their respective EIM.

Each of the EIMs is individually connected to the Supervisory Layer computer through fiber optic link in a star topology which has helped in reducing the overhead of address checking.

Software in EIM performs predetermined tasks either on request or on regular basis and confines to a group of instruments. For getting access to the equipment parameters, various proprietary command-response based communication protocols of different equipments have been implemented in EIM software. It periodically scans all equipments, connected to it, to read the status of their parameters and stores parameter values in its local buffer. It finally sends these values to the upper layer on demand. It also issues required control commands to the individual equipment for changing value of their parameters as desired by the operators. The task of handling data exchange between EIM and Supervisory Computer (SC) is performed following a customised protocol. The message received from SC is parsed and appropriate action is taken according to the command. Thus, dedicated software at this layer in coordination with upper layer provides total accessibility of equipment at remote control location.

Each EIM also handles safety interlocks associated with the subsystems it deals with and informs the interlock status to the upper layer.

The software for this layer is written in C language and after cross-compilation, the executable code is downloaded to the EIM controller.

Supervisory Layer (Layer-2)

The Supervisory Layer of DDACS has been realised by a system, developed using single board computer with Embedded XP operating system (OS) and named as Embedded Controller-1 (EC-1). The EC-1 (Fig. 5) is physically located at the cave of RIB facility, but away from the machine and interfaced with all the EIMs through fibre optic link to ensure necessary isolation and noise immunity. EC-1 performs the task of supervision by way of delivering the operator commands to the EIMs for controlling field equipments and receiving status information from them to report to the operators.

The Embedded Controller is built around a VIA processor operating at 800 MHz. However, the processor architecture is optimized for low power operation and reliable performance to meet the requirements of industrial applications. The embedded controller board supports operation over a wide temperature range of -20 to 70 deg. The controller runs on Embedded XP operating system which is built with minimum kernel resources, drivers and occupies minimum foot print when compared to the conventional Windows XP operating system. The OS, drivers and application software are embedded in a Flash memory. The board supports 512 MB of SDRAM and 512 MB of Flash memory. The controller board is operated from a DC supply of 5 - 28V DC. All the secondary DC supplies and the power management circuitry are built on-board. The Controller module has been tested and qualified for the environmental standard as per JS 55555 and Electromagnetic Compatibility (EMC) standard as per MIL STD 461.

The EC-1 system rack contains a Data Interface Unit (DIU) which physically connects EC-1 to all the EIMs. This unit mainly converts the fibre optic signals to the appropriate electrical signals and vice versa.

Main functions of this supervisory level controller are:

- To acquire data from EIMs (Layer-1), process the same and communicate the relevant data to control console at Layer-3.
- To receive the control messages from the control console, parse the same and distribute those to related EIMs to make effective on the equipments.

The application software, running in EC-1, is developed using Labwindows™/CVI and continuously sends commands to and acquires data from lower level EIMs. A customised command-response based protocol with 16 bit Cyclic Redundancy Check (CRC) has been adopted for reliable data communication between EC-1 and EIMs. Each equipment attached to an EIM is identified by unique "Instrument ID". This instrument id along with specific "Function Code" in a command message issued by the EC-1 uniquely define a particular operation (such as setting a voltage, reading a voltage, making a power supply on/off etc.) on a piece of equipment connected to a particular EIM.

Figure 5: Snapshot of Embedded Controller-1 System Module.

The EC-1 is connected upward to an Operator Interface Layer controller, named as Embedded Controller-2 (EC-2) through a pair of optical link. It receives operator commands and provides corresponding responses like transmitting a block of data or initiating the action to change the status or set the values of parameters of different equipments.

Operator Interface Layer (Layer-3)

Operator Interface layer handles operator interaction and provides a means of communication between operators and the accelerator machine. It mainly deals

with data representation and responds to each of the user actions. This layer has been realised by another embedded controller named as EC-2 and multiple PCs/ Workstations. These all are physically located in the RIB control room and form user friendly operator console.

EC-2 is similar to EC-1 in respect of hardware, but is functionally different. EC-2 is interconnected with different monitoring units (PCs/ Workstations) through a Local Area Network (LAN). All the status information of different equipments received from all EIMs by EC-1 are sent to EC-2 and finally displayed at PC/ Workstations. Any control action (for changing status of equipments) initiated at this layer finally reaches to the intended equipment through EC-1 and respective EIM. The response to user interaction has been kept within the human acceptable limits (500 milliseconds to 1 second).

Application software in EC-2 communicates with the supervisory layer embedded controller i.e EC-1 for data acquisition and control initiation. It communicates with the monitoring units using UDP protocol for displaying machine information in user friendly Graphical User Interface (GUI) (Fig. 6).

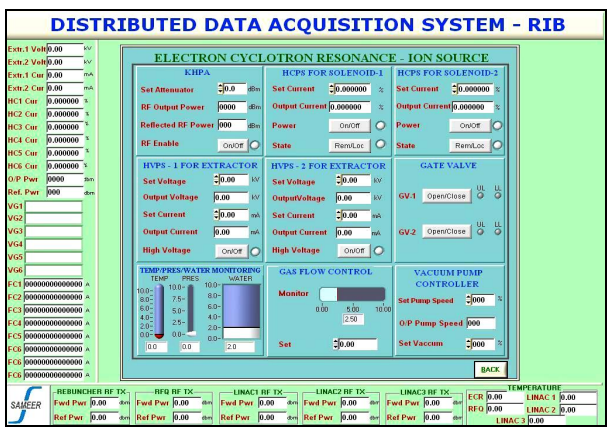


Figure 6: A typical GUI for ECR-IS Subsystem.

PRESENT STATUS

All the data acquisition and control modules have undergone functional performance check and EMI/ EMC qualification tests at Society for Applied Microwave Electronics Engineering & Research (SAMEER), Chennai before installation at RIB facility at VECC. Functional performance of the modules has been checked using a simulator system developed indigenously. Radiated Susceptibility test, Conducted RF Immunity test and Power Frequency Magnetic Immunity test have been carried out as per IEC 61000 standards on the modules to qualify them to be used in an accelerator environment. Presently, all the EIMs are installed and rigorous testing of the modules with the different sub-system has been carried out successfully at the RIB site. Embedded Controllers are also installed and thorough testing of those with EIMs produced expected results.

CONCLUSION

The RIB facility at VECC is under development and it is growing step-by-step towards its complete shape. The multilayer architecture of this data acquisition and control system is ideally suited for the RIB facility growing in phases. It has been envisaged to deploy a database server to be connected to the LAN at operator interface layer for data archival and off-line analysis.

ACKNOWLEDGEMENT

The authors would like to express their gratitude to Dr. R. K. Bhandari, Director, VECC and Dr. A. L. Das, Director, SAMEER for their constant support and encouragement for this activity. The authors also thankfully acknowledge the overall guidance and useful suggestions received from Dr. Alok Chakraborti, Head, Radioactive Ion Beam Facilities (RIBF) Group, VECC. All the officers and staff members of RIBF Group of VECC and SAMEER who are involved in this project deserve special thanks for their sincere efforts towards achieving the goal.

REFERENCES

[1] VECC, Kolkata, “Radioactive Ion Beam Project - An Overview”, July 2010.

[2] Pravin Fatnani, J. S. Adhikari, B. J.Vaidya, “Indus-2 Control System”, Proceedings of the Second International Workshop on Personal Computer and Particle Accelerator Controls (PCaPAC), January 12-15,1999 at KEK, Tsukuba, Japan.

[3] Brookhaven National Laboratory (BNL), USA, “Preliminary Design Report – National Synchrotron Light Source II”, November 2007.

CONTROL AND DATA ACQUISITION SYSTEMS FOR THE FERMI@ELETTRA EXPERIMENTAL STATIONS*

R. Borghes, V. Chenda, A. Curri, G. Gaio, G. Kourousias, M. Lonza, G. Passos R. Passuello, L. Pivetta, M. Prica, R. Pugliese, G. Strangolino
Sincrotrone Trieste, Trieste, Italy

Abstract

FERMI@Elettra is a single-pass Free Electron Laser (FEL) user-facility covering the wavelength range from 100 nm to 4 nm. The facility is located in Trieste, Italy, nearby the third-generation synchrotron light source Elettra. Three experimental stations (Fig. 1), dedicated to different scientific areas, have been installed in 2011: Low Density Matter (LDM), Elastic and Inelastic Scattering (EIS) and Diffraction and Projection Imaging (DiProI). The experiment control and data acquisition system is the natural extension of the machine control system. It integrates a shot-by-shot data acquisition framework with a centralized data storage and analysis system. Low-level applications for data acquisition and online processing have been developed using the Tango framework on Linux platforms. High-level experimental applications can be developed on both Linux and Windows platforms using C/C++, Python, LabView, IDL or Matlab. The Elettra scientific computing portal allows remote access to the experiment and to the data storage system.

INTRODUCTION

The newly-built free-electron laser FERMI@Elettra generated its first flashes of coherent light in the far ultraviolet in December 2010. A second phase of commissioning started in January 2011, with the goal of providing optimized FEL light to the experimental chambers [1]. The "first-shots" experimental activities, held in March and in July 2011, were mainly devoted to the commissioning and tuning of the three existing end-stations: LDM, EIS and DiProI.

The FERMI@Elettra fast data acquisition (DAQ) system is used to acquire, store and correlate multiple data sources. Based on the estimated scientific data throughput, the DAQ system has been designed in order to guarantee a data acquisition rate up to 1400 Gbyte/hour. During the experiment, scientific data is saved on a centralized storage system using a 10Gb ethernet connection.

Once stored, experimental data are available for offline data processing using the Elettra high performance computing clusters. Besides this, experimental data and local computing infrastructures can be remotely accessed through the Elettra scientific computing portal.

DATA ACQUISITION FRAMEWORK

FERMI@Elettra DAQ framework inherits from two years of experience gained during the machine commissioning and has been thought as the natural extension of the machine control system [2]. For every single laser pulse, at 50 Hz repetition rate, the DAQ system acquires, stores and links data coming from scientific instrumentation and machine diagnostics.

Control System Hardware Architecture

Dedicated scientific instrumentation apart, a uniform hardware architecture has been adopted for all the FERMI@Elettra experimental stations. According to Fig 2, at top there is a workstation dedicated to application software. Via a private VLAN, it accesses two low-level machines for interfacing instrumentation and a PLC dedicated to protection of the equipment.

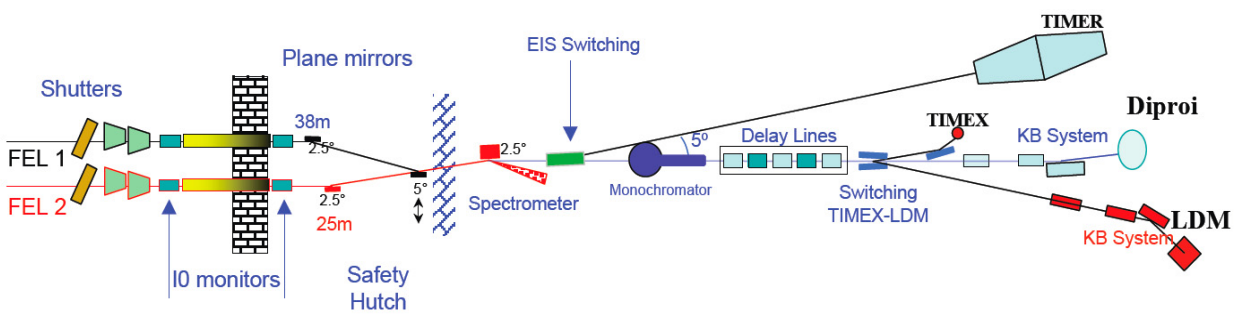


Figure 1: FERMI@Elettra beamlines layout.

*Work supported by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3

The workstation is an Intel Core i7-950 multi-screen desktop computer running either Linux or Windows.

Standard instrumentation is controlled by a VME system equipped with an Emerson MVME-7100 PowerPC board (Equipment Controller – EC). The EC also hosts the Event Receiver (EVR) by Micro-Research, that receives the machine bunch trigger via a fiber optics infrastructure. The EVR board can generate software interrupts and digital signals with a maximum jitter of 20ps. Instruments requiring special hardware interfaces or proprietary software are controlled and acquired by an Intel-based rack-mount server (Pentium XEON Quad Core). Both PowerPc and rack-mount Intel server are directly involved in real-time data acquisition. They are based on Linux (Kubuntu 10.04) with the Xenomai real-time extension [3].

Machine real-time diagnostics data are transparently shared among data acquisition computers using a network software application called Network Reflective Memory (NRM) [4]. The NRM system has been deeply tested during the commissioning period. It works on a fully dedicated, private network that covers the whole machine from the electron gun to the experimental hall.

Finally, a Siemens S7 series 300 PLC is used for the interlock system in order to protect equipment and devices from damage.

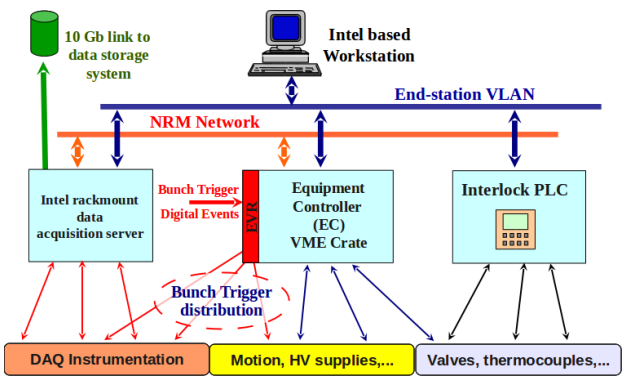


Figure 2: DAQ system hardware architecture.

Control System Software Platform

The DAQ system of the experimental stations is based on the Tango framework [5]. Low-level instrumentation control, scan implementation, data acquisition and analysis is done through Tango device servers. The low level device servers have been developed using C++, while the higher level ones have been written in Python and C++.

Each experimental station control system uses a private Tango database running on a virtual machine in the central system main server. Communication with the machine control system is made possible by special Tango devices that act as gateways. These are configurable servers that export specific attributes and commands between control system running on different VLANs.

As a general rule, automation and complex data acquisition operations are performed at Tango device server level, while client-side GUIs are used only for visualization purposes.

Graphical interfaces can be developed by beamline staff on Linux and Windows platforms using common frameworks like Labview, IDL, Python or Matlab. Applications developed by controls specialists have been written using QTango [6], a Qt based framework developed in house.

SCIENTIFIC DATA PIPELINE

Each laser pulse produced by FERMI@Elettra is unique in terms of spatial position, photon flux and energy spectrum. It is crucial to acquire and store *shot by shot* not only experimental data coming from the beamline detectors but also photon beam diagnostics data.

Every acquisition device server maintains and updates a circular memory buffer in which the acquired data are tagged with an increasing counter named *bunch number*. These low level devices export tagged data to the higher level storage device servers that organize experimental data in HDF5 (Hierarchical Data Format) [7] archives.

A challenging task in terms of data throughput is the acquisition of scientific detectors: until now only 1Mpixel detectors have been used but in the near future these will be upgraded to 4 Mpixel.

Table 1: Estimated Experimental Data Throughput

Endstation	FEL @ 10Hz	FEL @ 50Hz
DIPROI	140 Gbyte/hour	700 Gbyte/hour
LDM	280 Gbyte/hour	1400 Gbyte/hour
EIS	66 Gbyte/hour	330 Gbyte/hour

Data Storage and Analysis Resources

For each experiment a specialized Tango DAQ device performs data collection, online data reduction and storage in the HDF5 format.

Data archives are directly saved through the NFS protocol on a high speed centralized storage via a 10Gb ethernet connection.

The present centralized storage system is based on a 70 TB EMC Storage Area Network (SAN). Its next planned upgrade will provide FERMI users with a 1 PB distributed filesystem that will grant a data throughput of 1600 MB/sec.

Experimental data can be consequently analysed using the Elettra scientific computing infrastructure that currently includes:

- two state-of-the-art High Performance Computing Clusters equipped with InfiniBand and over 250 processor cores;
- an interactive computing cluster with about 40 processor cores;
- a cloud system, with a growing number of cores (presently 30).

The above resources can be remotely accessible through a specialized web portal: the Virtual Control Room (VCR) [8].

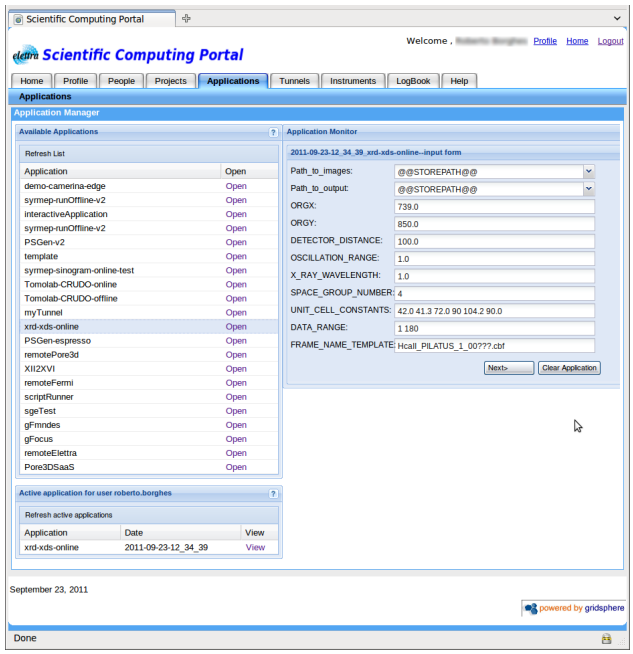


Figure 3: Elettra VCR web portal.

Virtual Control Room

The VCR is an open source web portal based on the GridSphere framework and Web 2.0 technologies. It is a highly configurable tool, designed for easy development and deployment of custom applications with transparent access to the underlying infrastructure. Among its main features are a safe tunnelling tool and an application manager (Fig. 3) that integrates a Jython-based scripting environment. The VCR acts as a front-end to the gLite-based Grids and to grid-enabled remote instrumentation via Instrument Element [9]. A number of tools like e-logbook, help-system and users-browser, make it a collaborative environment.

The portal provides secure remote operations of beamlines and experimental stations. It allows access to raw datasets and enables initiation of data processing during the acquisition phase. As such, the VCR is virtually suitable for any kind of scientific or business application and for the associated user communities.

CONCLUSIONS AND OUTLOOK

The FERMI@Elettra fast DAQ system has been tested during "first-shots" experimental activities held in March and July 2011. Working with a source of 10Hz repetition rate it has proven to be reliable, stable and dynamic enough for an experimental environment.

Further hardware and software upgrades will be carried out in the next months in order to handle a 50Hz repetition rate and 4Mpixel scientific detectors.

The efforts will be in accordance to the PANDATA ODI[10] related activities that include data policy, format standardisation, automatic metadata association and data management.

REFERENCES

[1] S. Di Mitri, "Commissioning and Initial Operation of FERMI@Elettra", IPAC 2011, San Sebastián, September 2011.
[2] M. Lonza et al., "Status report of the FERMI@Elettra control system", these proceedings.
[3] [3] http://www.xenomai.org
[4] L. Pivetta et al., "The FERMI@Elettra distributed real-time framework", these proceedings.
[5] http://www.tango-controls.org
[6] G. Strangolino et al., "QTango: a Qt Based Framework for Tango Graphical Control Panels", ICALEPCS 2009, Kobe, Japan, October 2009.
[7] http://www.hdfgroup.org
[8] http://www.dorii.eu/resources:adaptation:middleware:VCR
[9] http://www.dorii.eu/resources:adaptation:middleware:IE
[10] http://www.pan-data.eu

TRIUMF'S ARIEL PROJECT

J. Richards, D. Dale, K. Ezawa, M. Leross, K. Negishi, R. Nussbaumer, D. Morris, S. Rapaz, E.Tikhomolov, G. Waters

TRIUMF, Vancouver, Canada

Abstract

The Advanced Rare Isotope Laboratory (ARIEL) will expand TRIUMF's capabilities in rare-isotope beam physics by doubling the size of the current ISAC facility. Two simultaneous radioactive beams will be available in addition to the present ISAC beam. ARIEL will consist of a 50 MeV, 10 mA CW superconducting electron linear accelerator (e-Linac), an additional proton beam-line from the 520MeV cyclotron, two new target stations, a beam-line connecting to the existing ISAC superconducting linac, and a beam-line to the ISAC low-energy experimental facility. Construction will begin in 2012 with commissioning to start in 2014. The ARIEL Control System will be implemented using EPICS allowing seamless integration with the EPICS based ISAC Control System. The ARIEL control system conceptual design and initial results from a prototype injector control system will be discussed

INTRODUCTION

Commencing in 2012, with staged installations, the TRIUMF ARIEL project will triple TRIUMF's Rare Isotope Beam (RIB) program by providing two new radioactive beam sources. Combined with the existing beam from ISAC, TRIUMF will have three simultaneous radioactive beams available for experiments or beam development. The ARIEL project consists of a superconducting electron linac, a new proton beam line from the TRIUMF cyclotron, 2 new target stations, 2 new mass separators and a low-energy beam line switchyard feeding into the existing ISAC accelerators (Figure 1). The project implementation will extend over 10 years.

ARIEL Phase I, to be completed by 2015, consists of the construction of a RIB Factory comprised of a new RIB driver, an electron beam line, and a temporary production target. The driver, a CW superconducting electron linac with 250 kW beam power operating at 2 K will use a demonstration dump. ARIEL Phase II will complete that factory with RIB production targets, a linac energy upgrade, and the addition of a proton-driven RIB production line.

The Phase 1 electron linear accelerator (e-linac) will be comprised of a 300 keV Thermionic gun (e-gun), an injector cryo-module with a single 9-cell cavity and one accelerator cryo-module with two 9-cell cavities. The injector provides a 5-10MeV beam current of 10 mA, i.e a beam power of up to 100kW. The accelerator cryo-module will boost beam energy to 25 MeV. The division into injector and main linac accelerator allows a possible future expansion for energy recovery or energy doubling.

The ARIEL project is aided by collaboration between TRIUMF and VECC of Kolkata, India, which is planning a RIB facility with an e-linac similar to ARIEL. The injector cryo-modules are being jointly developed and one module each will be built for VECC and ARIEL. A beam test for the VECC module is scheduled at TRIUMF in 2012. This presents an opportunity to prototype many elements of the ARIEL control system.

Figure 1 shows the layout of TRIUMF's RIB facility indicating the additions by the ARIEL project.

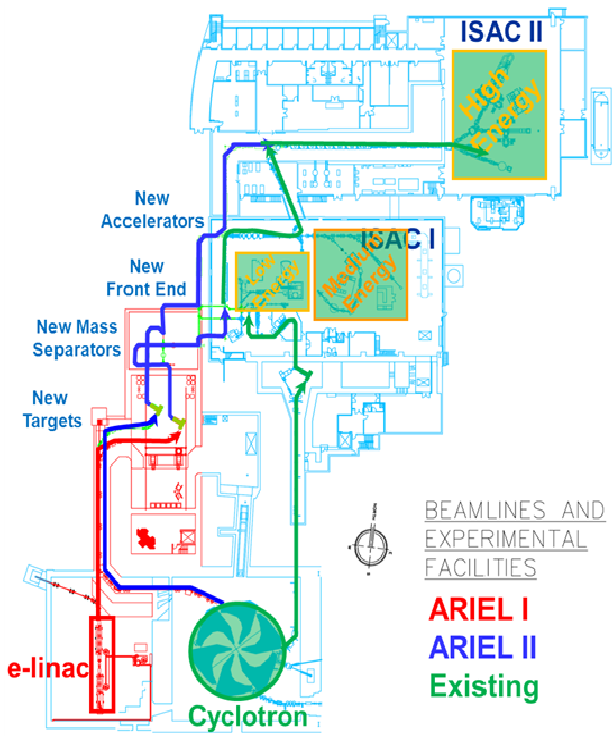


Figure 1: TRIUMF's ARIEL Project.

CONTROL SYSTEM: CONCEPTUAL DESIGN

The ARIEL Control System (ACS) will be implemented using the EPICS toolkit relying on established technologies and productivity tools of the ISAC Control System [1] wherever possible:

- The IOC software will be designed using the tdct [2] EPICS database configuration tool using the "object-like" device and component hierarchy developed for ISAC
- The ISAC device relational database web tools [3] will be used to:

- capture device information and interlock specifications
- instantiate devices in tdct schematics.
- validate interlock implementation in PLCs against the specification.
- auto-generate device control panels with interlock visualization [4],[5]

Evolution of the ISAC standards will be driven by the fact that high power beams are controlled and the need to further optimize resource usage in the control group.

In order to reduce the knowledge base to be maintained within the controls group, Linux will be used as the sole operating system on all IOC target architectures, servers and operator interface stations [6].

Integrated operation of the Cyclotron, ARIEL and ISAC control systems is not needed until after 2015 – when the proton beam line 4North comes on line. For this, a relocation of the Cyclotron and ISAC control rooms together with the ARIEL controls into one operations centre is being considered.

Machine Protection System

With a Phase 2 capability of 500 kW C.W. of beam power, the e-linac has been referred to as an electron blow torch. Therefore – in addition to the ISAC model - a fast machine protection system (MPS) will be required which is intended to prevent the high power beam from damaging the ARIEL accelerator and electron beam line. The MPS conceptual design is based on the JLAB implementation and intends to use JLAB–designed hardware that interface with photomultiplier tubes and ionization chambers [7]. The MPS will consist of a reactive fast shutdown system, a beam loss monitor system in support of the fast shutdown, and a preventive status monitoring and veto system. The MPS has unrestricted control of the e-gun grid modulation and HVPS with reaction time < 10 μsec in event of a full point loss. The MPS independently monitors devices deemed critical to the MPS function.

The MPS will use a modal view of facility operation. An e-linac Operating Mode is a configuration of beam characteristics (current, energy) and machine destination modes that permit such beam (low energy beam dump, high energy beam dump, etc). The desired operating mode is selected by the operator, but may be "downgraded" by the MPS based on events or operating conditions. The MPS will inform the control system of the actual Operating Mode which will form part of the control system interlocks for critical devices. Once a valid beam mode is established, the MPS system will automatically configure itself to monitor only those inputs which are required for beam operation to the designated beam dump.

An MPS trip will result in a hardware distributed event signalling the control system to acquire a time-stamped, beam-mode relevant, device snapshot. This will allow MPS trip post mortem analysis.

Operator interface to the MPS will be through the EPICS control system. Operator screens will be used to select machine configuration, monitor system status and reset trips. Additional screens may be used to set up beam loss monitors, initiate built in self-tests of fast abort components, and analyse trip data sets.

RF

EPICS will communicate with HPRF systems for control and to obtain device status, diagnostic values and interlock status. Interlocks for the HPRF systems will be directly supplied by the control system PLCs. The power supply for the 30 kW Inductive Output Tube that feeds the injector cryo-module will be controlled using the unit's RS232 interface and Streamdevice.

The 1.3 GHz, 300 kW klystron needed for the accelerator cryo-module has now been ordered from CPI. Tendering for all hardware and software to control the operation of the five required HV power supplies is underway. The contract stipulates PLC control for voltage regulation and protection of both the supplies and the klystron. The control module will present a Man-Machine Interface (MMI), a state machine for implementing the sequences required for proper operation of the power supplies for the klystron, and a latching interlock system that provides time-stamped fault logging. An interface layer for EPICS command and readback is stipulated since remote control of the HPRF sources via the control system is required.

Integration of the klystron power supply system and the klystron RF system is included in the scope of work as an option of the tender. If not provided by the vendor, this integration which encompasses interlocks, control, and monitoring of the complete RF system will be implemented in-house. The RF system includes the klystron, circulator with waveguide directional couplers, high power dummy loads with associated cooling and temperature monitoring systems, and a window arc detector. The operating parameters of the klystron for achieving 300 kW CW RF power at the dummy load are yet to be established. The response time of the interlock system must ensure protection of the klystron and other system components under any fault condition. Interlock trips must be time-stamped at a resolution of 1 ms or better. This precludes the use of a slow PLC scan. The status of all signals at the time of fault must be readable via MMIs and EPICS interface.

Figure 2 shows the e-linac RF components as they will exist for Phase 1 in 2014.

Low level RF is controlled by a dedicated RF control system [8] running on PCs running Windows. This local control subsystem is integrated into the EPICS control system using a shared memory interface with a "soft" IOC [9].

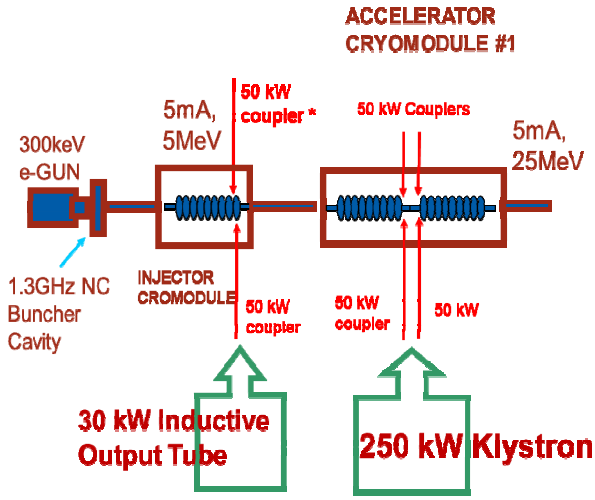


Figure 2: E-linac RF – Phase 1.

Beam Diagnostics

The E-Linac Diagnostic design has already been the subject of an International Review. Expected prompt radiation effects near the beam line would require all electronics to be rad-hard. No electronics will be installed in the E-linac hall, except in select shielded areas. This imposes restrictions on the type of instruments and bus lengths used since a 50 meter cabling path will be required.

The majority of e-linac diagnostic devices are slated to use VME for analog control and read back using 16 bits precision. The response time requirement of the control system is 10 Hz with a 5 Hz rate for updates on the operator consoles. Diagnostics that have data acquisition rates faster than 10 Hz will be handled by specialized hardware which will provide a first order processing. Data communication to the control system may then follow shared memory methods established in the ISAC control system for "soft" IOC data acquisition [9].

Diagnostic elements include button-type beam position monitors (BPM), ring capacitive pickups, DC current transformers, camera screen devices and wire scanners. The control of screen devices is being provided by the University of Victoria, a joint partner in the E-Linac project.

Device actuation into the beam line will be interlocked by the MPS subject to a proper facility operation mode (beam energy, beam intensity) for the device. Device trips, under conditions in which the device should be protected from the beam power, must be dealt with using "upstream" protection since the diagnostic devices cannot move fast enough to protect themselves. Device radiation dosage history will affect the performance of some units. Hence records must be kept by the control system and alerts issued when designated thresholds are reached.

Beam Optics

In ISAC, the supplies powering electrostatic or magnetic beam optics devices were controlled by individual TRIUMF-developed microcontroller cards and supervised from the IOCs via CAN field-bus [10]. This solution, while cost-effective and low maintenance at ISAC, may not be propagated to the electron beam optics control because of a different cost situation and concerns over reliability in the high power beam environment. At present, different approaches for low current and high current power supplies are being evaluated.

For optics devices requiring no more than 10V/5A, the Matsusada R4K-80L is a strong candidate for its cost effectiveness (150+ units are required). A control solution where small R4K-80L clusters are controlled from an EPICS IOC using ASYN/Streamdevice via RS232, with interlocking and external polarity switching provided by PCI digital I/O cards, will be evaluated during the VECC test phase using x-86 IOC targets. [6].

For optics devices requiring more powerful controllers, three possible methods of control are being considered: RS232, CANbus, or PLC. The criteria for instrumentation bus selection includes how fast the PS can be controlled, reliability of power supply shut off, and cost. In each of the following control scenarios, PLC interlocks on magnet temperature switches will cut power to the device. For an RS232 controlled supply, it is estimated that a polling rate of 10 Hz could be maintained from a Linux IOC. If the unit failed to respond, a method to remotely turn off the supply could be implemented using PCI digital I/O driving a relay which removes the power. The design of existing TRIUMF CAN-bus microcontrollers provides for a maximum update rate of 10 Hz and uses a beacon to determine communication health, resulting in a power supply being shut off on beacon loss. PLC control of direct analog/digital supplies is already a feature of the ISAC control system for units at elevated potential. The PLC response rate, reliability, and precision are satisfactory – however price factors may limit this choice.

Source E-GUN

Tests using a 100 keV electron gun have already been performed on a prototype using the standard ISAC Controls model: PLC, VME, and CANbus. Additional tests will be conducted using this source before it is replaced with a 300keV e-gun for VECC injector cryo-module tests.

Vacuum System

As in ISAC, control and interlocking for vacuum devices will be implemented using PLCs and the interlock implementation will be automatically verified against the specification [4]. Schneider/Modicon PLCs of the Quantum and M340 series are currently being commissioned in the VECC injector beam line test. The PLCs are supervised by EPICS using a TRIUMF developed Ethernet driver with the Modbus over IP protocol.

A major departure from the ISAC vacuum system design is the use of movable pumping stations for the ARIEL vacuum system. This translates into large cost

savings as turbo and scroll pumps are mounted on carts and connected to sequential vacuum volumes as required. Ion pumps maintain the attained vacuum in the beam line when the pumping cart is detached.

Cryogenics

The cryogenic plant for the e-Linac is currently in the tendering process. The Helium cryogenic system has three components: 4K liquid He in a closed re-liquefaction – refrigeration loop, 2K liquid He produced within the cryo-module by sub-atmospheric pumping, and room temperature gaseous He. A 77K liquid Nitrogen system provides for pre-cooling of He gas. The design places strong emphasis on monitoring and maintaining the helium purity. The scope of work mandates a PLC implementation with a documented TCP/IP interface to EPICS. The existing cryogenic system in ISAC has EPICS controls implemented in-house [11].

High Level Applications

E-linac tuning will require Control System High Level Applications (HLA) based on physics models. The XAL [11] framework, which integrates with the EPICS control system and provides for accelerator models, was chosen as a platform for HLA development, testing, and execution. XAL uses XML descriptions of the beam transport model that include specific optic and beam monitor control system process variables. TRIUMF XAL developments have thus far focused on low energy beam transport models. An empirical model has been created which has been proven to perform correctly at 300 keV [13].

HVAC

The buildings that house the ARIEL project will have a commercial building control system supporting the BACnet standard. In many situations the need to monitor such parameters as room temperature or air flow for correlations with beam or machine behaviour is required. BACnet/Ethernet is the physical and datalink layer that is proposed for the ARIEL facility. The building controls data can be accessed using BACnet support for EPICS [14].

REFERENCES

[1] R. Keitel et al, “The ISAC Control System – Phase II”, ICALEPCS05, Geneva, 2005
[2] R. Keitel, “TDCT- A configuration tool for EPICS runtime databases”, ICALEPCS09, Kobe, 2009
[3] R. Keitel, “Generating EPICS IOC Databases from a Relational Database – a Different Approach”, ICALEPCS01, San Jose, 2001
[4] R. Keitel and R. Nussbaumer, “Automated checking and visualization of interlocks in the ISAC Control System”, ICALEPCS01, San Jose, 2001
[5] R. Keitel, “Edlbuild – display generation for the EPICS EDM display manager’’, ICALEPCS05, Geneva, 2005
[6] J. Richards, R. Nussbaumer, G. Waters, S. Rapaz”, ISAC EPICS on Linux: The march of the penguins”, ICALEPCS11, Grenoble, 2011
[7] J. Yan and T.L. Allison, “Signal processing board for new beam loss monitor”, ICALEPCS11, Grenoble, 2011
[8] K. Fong, M. Laverty, S. Fang, “RF Control System for ISAC II Superconducting Cavities”, EPAC2002, Paris, 2002.
[9] E. Tikhomolov, “Interfacing of peripheral systems to EPICS using shared Memory”, ICALEPCS07, Knoxville, 2007
[10] D. Bishop, D. Dale, H Hui, J. Lam, R. Keitel, “Distributed Power Supply Control Using CAN-Bus”, ICALEPCS97, Beijing, 1997
[11] R. Nussbaumer et al, “Cryogenics controls in the ISAC-II Superconducting RF Accelerator”, ICALEPCS09, Kobe, 2009
[12] http://www.ornl.gov/~t6p/Main/XAL.html
[13] Y. Chao and G. Goh, “Low-Beta Empirical Models used in Online Modelling and High Level Applications”, IPAC 2011, San Sebastian, 2011
[14] R. Nussbaumer and G. Waters, “BACnet support for EPICS”, ICALEPCS05, Geneva, 2005

UPDATE ON THE CENTRAL CONTROL SYSTEM OF TRIUMF'S 500 MeV CYCLOTRON

M.M. Mouat, E. Klassen, K.S. Lee, J.J. Pon, P.J. Yogendran, TRIUMF, Vancouver, Canada

Abstract

The Central Control System of TRIUMF's 500 MeV cyclotron was initially commissioned in the early 1970s. In 1987 a four year project to upgrade the control system was planned and commenced. By 1997 this upgrade was complete and the new system was operating with increased reliability, functionality and maintainability. Since 1997 an evolution of incremental change has existed. Functionality, reliability and maintainability have continued to improve. This paper provides an update on the present control system situation (2011) and possible future directions.

INTRODUCTION

The Central Control System (CCS) has been running almost continuously since 1974 when the first beam was extracted. Since those early days, and still ongoing, there has been a constant, incremental evolution of the CCS. These advances are punctuated periodically by more major changes. Even during major changes of the control system, which may occur during multi-month shutdown periods for the cyclotron, the scheduled off time for the CCS has been limited to durations of not more than a few days because many groups on site require CCS functionality even when beam is not being delivered. Instances of shutting down the CCS rarely occur now.

Evolution of the CCS has been shaped in part by constraints such as the need for availability and accommodating these constraints has presented interesting challenges. An example of this evolution is the emergence of two, symmetric, computer clusters, one for software development (and diagnostics) and one for production operation. The Development Cluster can provide all of the functionality of the Production Cluster. During a major, multi-month, site shutdown, disruptive upgrades such as that of the fibre channel storage system on the Production Cluster can proceed while the CCS software runs with full functionality in the Development Cluster. This is an important redundancy.

Almost all applications in the CCS use the X Window protocol to display their output. Two more clusters of computers (the Display Clusters) are dedicated for display purposes. Within the Display Clusters, 16 multi-headed workstations are used simply to receive output from the servers, display the X window on monitors, and allow keyboard and mouse input from the user. These workstations do not run applications.

During the life of the CCS, its computers have evolved from the initial 16 bit processors, through 32 bits to the present blend including multi-core, multi-processor 64 bit computers. Disk storage changed from the original fixed head devices, through removable cartridge styles, to multi-access DSSI drives, to early fibre channel, and now virtualized fibre channel is being installed. There was a time with no network, then the network started with collision-based 10 Mb/sec simplex thickwire, and moved through thinwire, to FDDI, to 10/100 Mb/sec switched Ethernet, and presently to the 1 Gb/sec configuration.

There are a number of general goals that most controls systems will strive to meet, characteristics such as reliable operation (low downtime), ease of new developments and maintenance, a high level of functionality, longevity while providing the required functionality, simplicity of use, and the like. The experience of TRIUMF's Cyclotron Controls Group is that to identify and understand the goals takes a cooperative approach with the end users, within the Controls Group itself, and with management. The end users are typically cyclotron sub-system groups (Vacuum, RF, Magnets, etc) and especially the Cyclotron Operations Group, but include a number of other groups such as Beam Development and ISAC Operations.

HARDWARE STRUCTURE

The present hardware configuration is a natural evolution from the initial design, which was destined to be very effective. This design employed CAMAC as a data acquisition and control bus with multiple, hardware arbitrated, computer connections (the GEC Elliott Executive crate). The configuration is referred to at TRIUMF as multiporting. This concept of multiporting continues to be a major characteristic and powerful advantage within the CCS architecture. Having multiple computers able to independently, efficiently, and (effectively) simultaneously have access to almost all CCS device channels, largely eliminates the need, overhead and complexity of passing commands and data between computers (IOCs and servers). The CCS does not have IOCs. There are only servers and these servers are capable of quickly accessing parameters across the breadth of the cyclotron from the ion source to the beam dump. This allows functionality such as quick, high level interlocks that otherwise would be difficult to provide.

CAMAC is a set of well defined, open, standards with many useful features such as a simple inter-crate bus system, interrupt handling, multiple topologies, locally arbitrated intelligence in a crate, etc. CAMAC is now seen by many people as an old technology (of approximately the same age as VME). At TRIUMF the CAMAC continues to function very well. The CCS's CAMAC was incrementally expanded to the present configuration, which has multiple, independent systems running in parallel, with each system capable of supporting up to seven parallel or serial branches. Each of these systems is capable of interfacing multiple computer connections, presently providing between three and eight computer connections.

The electronics in the CAMAC varies from the original (1970s), to the very recent (2011). Many of the new CAMAC modules (ADCs, DACs, etc.) are newer in fabrication, of a more recent design, of better function, higher resolution, and of better performance than our new PLCs, and at a lower cost per channel. Although design and fabrication of CAMAC electronics is easy by today's standards and witnessed by recent modules, industry support of CAMAC is in sharp decline.

Other data acquisition and control connections are also supported, including links such as Ethernet to PLCs, Ethernet to other processors, Ethernet to terminal servers (for serial ASCII), Ethernet to specific devices (such as an arbitrary waveform generator), Ethernet to web servers (such as a weather station), and VME.

PLCs are being increasingly used in the CCS because of their ease in interlock implementation, modularity, and widespread acceptance. There are still more than 10, old, microprocessor-based, local control and interlock systems that communicate with the CCS and are in need of replacement. Refurbishing each of these systems represents a significant amount of monetary, staff and time resources, all of which are in short supply. Projects such as replacing the main tank vacuum microprocessor system continue to be deferred. The original systems run with remarkable reliability, although two years ago a microprocessor failed and integrated circuit chips were found with chip leads corroded away.

The CCS network expansion has a variety of characteristics including aspects such as in the number of connections, in bandwidth and latency performance, in functionality, in complexity of management, in the resources required for maintenance and development, and in the resources required for security. To help address security, a security-in-depth (layered) approach is taken. This includes items such as firewalls, multiple VLANs, individualized computer security filtering, and monitoring. Diagnostic devices such as the Fluke OptiView Series III network probe has proven to be particularly useful in characterizing network activity and in diagnosing network problems (such as to identify multiple use of an IP address).

Hardware diagnostics have increased significantly, especially in specifically targeted areas. Each CAMAC crate is now equipped with a specially designed diagnostic module (60 in all), which is queried every six seconds to identify early signs of power supply issues, temperature issues (ambient and fan failure), and in the future other potential problems will be monitored.

The widespread expansion of Uninterruptible Power Systems (UPS) within the CCS has boosted reliability and uptime. Although various sub-systems suffer during power issues, normally the CCS now runs through these events and helps to identify and analyze the consequences. This is particularly helpful for Operations.

The main console has been tailored to work efficiently for the Operations Group. Approximately 35 dedicated CCS monitors of varying sizes from 18 to 42 inch are driven by two clusters of multi-headed workstations. Twelve of the workstations are in the main console.

In addition to workstations, the console still provides knobs, buttons, and analogue meters to enhance the tuning ergonomics. These devices come in both dedicated and assignable instances and the underlying software support provides a number of enhanced features such as configurable beam current run-up characteristics to take a beam-off situation to the desired beam current.

SOFTWARE (DEVELOPMENT & PRODUCTION) ENVIRONMENTS AND FRAMEWORK

The Central Control System software was first created (early 1970s) using the recently developed NATS (Nova Asynchronous Tasking Supervisor) realtime operating system. The control system software was written in assembler and distributed across a number of Data General Nova computers. These Nova computers ran, with daily reboots due to application bugs, for many years until the completion of an upgrade project that was initiated in 1987. The choice was made to move to Digital Equipment Corporation VAX computers running the VMS (also called OpenVMS) operating system but to maintain the data acquisition hardware infrastructure, which was predicated on CAMAC. Due to the constraints of the upgrade, the conversion was not completed until December 1996. Although the Cyclotron Controls Group also supports Linux, Microsoft Windows, and Solaris, VMS is the operating system underlying the cyclotron's production software. VMS runs efficiently and reliably. As the computer hardware evolved from 32 bit VAXes to 64 bit Alphas and then Itaniums, the necessity arose to simultaneously support applications and software infrastructure on multiple hardware architectures. The OpenVMS development and runtime environments make this issue straightforward. There is only one code base for the CCS and users are unaware of the hardware they are running on. Now, only Alphas and Itaniums are supported but adding another architecture should be simple.

Programming, debugging and software testing is done on the Development Cluster. When the software is deemed to be running correctly it is moved to the Production Cluster for deployment.

A master Oracle database is used to maintain device and acquisition information. From the master database a single, static, runtime database with all necessary device and acquisition information is extracted and normally moved to each of the computers doing data acquisition and control. Configuration management is effectively eliminated because there is only one current configuration and normally all of the computers have it. A new version of the runtime database can be installed on the fly, without needing to reboot or stopping any applications that are running. Only newly started applications will dynamically link to the new version, thus multiple versions may simultaneously be active in each computer.

This flexibility allows new versions to be tested while the latest commissioned version continues to run.

OpenVMS, like Linux and Unix, employs the X Window protocol for display purposes. The vast majority of display processes use X Windows but web based applications are supported. Web based programs such as Oracle forms and a Safety System program for testing gamma and neutron monitors are frequently used.

The main Operations console hardware is tied into the CCS via interrupt driven CAMAC signals that go to the OpenVMS servers. Software detects device selection and parameter modification (such as an increment via button press) so that dedicated applications can quickly and appropriately adjust the accelerator equipment and the X window displays. Input via X windows is also available.

The CCS framework has mostly been developed in house but there are some commercial products such as the graph widgets and message bus infrastructure.

A communications link exists between the CCS and the Central Safety System (CSS). In the CCS a touch panel and associated monitor provide Operations and others an interface with the CSS. This system has 39 display pages. In addition, there are a number of other display pages showing CSS data, such as neutron monitor levels displayed as vectors, which change colour depending on their value as compared to the trip and warning levels.

A recent addition to the Cyclotron's control system is the implementation of EPICS for a part of the ion source and injection line's vacuum equipment. This work has been structured to follow the standard EPICS implementation as used in ISAC, a radioactive ion beam (RIB) facility at TRIUMF. The Cyclotron Controls Group has previously been involved in implementing and supporting EPICS on secondary beamlines for experimental facilities.

Access directly from an application in a server to production equipment connected to the CAMAC takes approximately twenty microseconds (depending on computer, command, and location of the equipment). The hardware component varies between three and eleven microseconds (with an average of five uS). A new hardware interface, which is in design, should lower the access times in both hardware and software. The effective bandwidth is also increased because there are multiple CAMAC systems and computers running in parallel.

Graphical displays in the Main Control Room that are generated from CCS applications can be roughly placed into one of two categories, displays that run continuously in a specific location for all operational shifts (the vast majority), and displays that change from shift to shift and are started and used dynamically as the need arises. Many LCD monitors have multiple application displays of the first category running on them. All of these applications normally run on just one server and if a full startup is requested, approximately 180 applications (and displays) are created during the startup period. A full startup has been slowed down (to about five minutes) to provide programmed delays for sequencing the window startups. The displays are positioned and sized to nicely tile the monitors' screens. If the Controls Group wants to shutdown the active display server, with one command executed on another server all of the applications/displays can be started and run simultaneously to those on the first server. The previously existing displays will be covered over by a new copy of the application's output and the first server can be shut off without Ops losing any displays. The configuration of displays is handled by Ops.

APPLICATIONS

The CCS has five primary applications that play major roles in cyclotron operation: XTpages, Xstrip, Histograms, Scans, and Elog. The key application is called XTpage (originally for X window Terminal Pages). This is more a framework with a common user interface for what is currently 325 display pages. Each display page is an application on its own. Emittance scans, target scans, control of probes, individual device control, loading injection line tunes, viewing and controlling sub-system parameters, bypassing PLC interlocks, control system diagnostics, and many other activities are interfaced via XTpages. In addition, each page has a user customizable help page. XTpages follows the CCS colour rules and provides a standardized user interface.

Xstrip is a heavily used strip chart tool. On each instance, the user can retrieve up to ten years worth of logged data on up to twenty parameters and plot the data. The tool will let the user then go into active mode and append current data in realtime at up to 20 hertz. Xstrip has received, and continues to receive, a significant amount of ongoing development to add features. Many users and especially Operations have contributed to its characteristics and success. By right clicking on an active parameter on an XTpage, you can start an Xstrip application, automatically retrieve/plot that parameter's logged data, and then start gathering active data.

The Histograms are a family of vector plot applications. Plothist, Plotcross, Complex Plot, etc will allow users to represent parameters as vectors in realtime.

The Scans is a server based package of event handling software that loads pre-defined scripts and then runs in realtime. There are two basic forms, event handling that provides messages only, and event handling that provides messages and device interlocking. Because of the CCS architecture, the Scans can quickly read parameters across the breadth of the CCS, draw conclusions and take actions. The Scans are modelled on PLC operation but provide the power and flexibility expected in a server.

The Elog is a web based, database backed, electronic log application. Other CCS applications have been modified to support saving graphical displays for inclusion in the Elog.

NEW/INCREASED FUNCTIONALITY

Only a few of the developments since the last Status Report [1] will be mentioned here. The targets in RIB facilities are important and need stable incident beams. Because beam over current and position errors may cause

machine protection issues, beam trips are periodically initiated. To enhance beam stability, beam current (amplitude) feedback, and beam position feedback loops have been implemented [2]. The feedback loops have significantly reduced these trips. In addition, a procedure referred to as "soft trips" is now used [2], which allows the beam current to be quickly and iteratively reduced to avoid the trip situation as opposed to just turning the beam off. This "soft trip" is much easier on the targets, and probably other equipment such as extraction foils. In another ISAC related development, an application was setup to rotate the proton beam in a circle on the RIB target to assist in beam and target development.

There have been a variety of enhancements to existing applications (XTpage, Xstrip, Scans, and Histograms). In the cases of emittance and target scans, isolated applications were incorporated into XTpages. Online diagnostics have been enhanced. One major step forward, as described earlier, was the deployment of a specially designed diagnostic module and initial software support. In what is an ongoing project, the Oracle X window based forms are being replaced by web based forms.

RELIABILITY, PERFORMANCE, MAINTAINABILITY

The Central Control System runs reliably. Figure 1 shows that since the upgrade was started in 1987, the control system downtime has been decreasing to the point now where it varies according to the (small) number of hardware failures. Equipment is normally spared at an appropriate level, which helps maintainability. And most failures can be analyzed rather quickly with the aid of improved diagnostics. Online diagnostics and daily checks often show problems before downtime occurs.

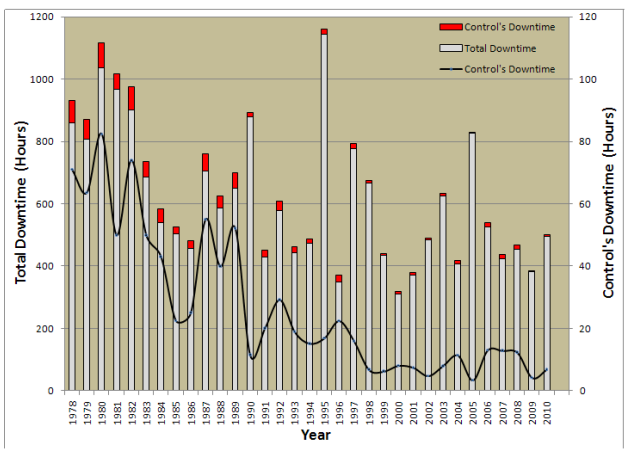


Figure 1: Cyclotron Total and CCS Downtime.

Reliability, performance and maintainability have all benefited from new designs and new electronics. New DACs and ADCs have been deployed in the last few years and new digital I/O is being prepared.

PROJECTS DONE, PROJECTS TO DO

The Cyclotron Controls Group inherited responsibility for the hardware of more than twenty old microprocessors. Some of these systems are obsolete and do not need replacement. Two systems, the Solid Target Facility and Beamline 2C's vacuum system were each replaced by a PLC. There are still more than ten microprocessors, involving major systems such as the main tank vacuum and ISAC target protection, which need to be replaced. The scope of this work is daunting.

A large project involving a major upgrade of the vertical section of the ion source and injection system was recently completed. Control of the vacuum component of this work was done using a PLC and EPICS.

FUTURE

The future will continue to bring the incremental changes as seen in the past. Changes to the fibre channel storage arrays are now underway. The site network is almost always being upgraded, with significant redundancy plans established. Enhanced Scans are also underway. These changes will provide better efficiency, clarity, and flexibility. Upgrades of the microprocessor systems will occur but firm plans are still to be made, although almost certainly PLCs will be used.

SUMMARY

In summary, the Central Control System is running well, providing good functionality, performance, and reliability, while remaining efficient to support. Many of the lessons learned must be viewed knowing that TRIUMF is a lightly funded facility relative to many similar sites. The CCS has done well to stay with technologies that are reliable and that have an evolutionary path. The VMS, CAMAC and multi-access disk storage systems are good examples. The effort to recognize and remove less reliable/harder to support technologies has been rewarded. Investing in diagnostics and tools to detect problems early on before downtime occurs or more serious consequences follow, has been worth the effort. The perspective of taking ownership of the core infrastructure, to maintain adequate spares, and to follow the philosophy of don't change for change's sake, change to gain something, has worked well.

REFERENCES

[1] M.M. Mouat, B. Davison, S.G. Kadantsev, E. Klassen, K.S. Lee, J.E. Richards, T.M. Tateyama, P.W. Wilmshurst, P.J. Yogendran, "Status Report on the TRIUMF Central Control System", ICALEPCS1997, Beijing, Nov. 1997, p.43 (1997).

[2] J.J. Pon, E. Klassen, K.S. Lee, M.M. Mouat, M. Trinczek, P.J. Yogendran, "Recent Changes in the 500 MeV Cyclotron's Central Controls System to Reduce Beam Downtime and Beam On/Off Transitions", ICALEPCS2009, Kobe, October 2009, TUP043, p. 179 (2009); http://www.JACoW.org/

THE GATEWAYS OF FACILITY CONTROL FOR SPRING-8 ACCELERATORS

M. Ishii#, T. Masuda, R. Tanaka, A. Yamashita
JASRI/SPring-8, Hyogo, Japan

Abstract

Utilities data such as the air temperature in a machine tunnel, power line voltage, and temperature of machine-cooling water are managed by the facility management system. Previously, the accelerator control system could not obtain most of the required utilities data because the accelerator control system and facility management system were independent systems without an interconnection. In 2010, we updated the old facility management system. In order to facilitate data acquisition, we constructed gateways between the MADOCA-based accelerator control system and the new facility management system, installing the BACnet protocol. The data acquisition requirements were as follows: to monitor utilities data with the required sampling rate and resolution, to store all of the acquired data in the accelerator database, to maintain independence between the accelerator control system and the facility management system, to ensure a future expandability to control the facility equipment from the accelerator control system. We outsourced the gateway construction, including the MADOCA-based data acquisition software, to solve the problems of limited manpower and short work period. In this paper, we describe the system design and the outsourcing approach.

INTRODUCTION

Utilities data such as the air temperature in a machine tunnel, power line voltage, and temperature of machine-cooling water assist accelerator specialists in studies on the correlation between beam stability and environmental conditions. In SPring-8, the accelerator control system, which used a Message And Database Oriented Control Architecture (MADOCA) framework [1], and the facility management system, which used the old MELBAS (Mitsubishi Electric Building Automation System), were independent systems with no interconnection. For over 15 years, the accelerator control system had no provision for obtaining most of the utilities data handled by the facility management system, except for that from legacy and offline devices such as paper records, floppy disks, and USB memories.

We started to implement an update of the old facility management system in June 2010, and we completed the project in March 2011. The new facility management system employs BACnet [2] as the underlying protocol. BACnet is an open protocol of data communication for building automation and control networks, and it has been standardized by an ASHRAE, ANSI, and ISO. The BACnet protocol defines a number of data link/physical layers, including Ethernet and ARCNET. BACnet is widely used in the building management market. We built the gateways to facilitate communication between the MADOCA-based accelerator control system and the new facility management system.

To implement these gateways, we had to overcome the problems of limited manpower and short work period. We outsourced the construction of the gateways, including the data acquisition software of MADOCA framework. We describe the system design and outsourcing approach below.

SYSTEM DESIGN

In this project, our objective was to make a provision for the accelerator control to acquire the utilities data handled by the new facility management system. The RF conditions are very sensitive to the facility equipment and environmental conditions such as the air temperature of machine tunnel and temperature of the machine-cooling water. We focused only these utilities data and constructed a utilities data acquisition system (utilities DAQ system). The requirements for the system were as follows:

- The utilities DAQ system can acquire the utilities data for the facility equipment with the required precision and sampling rate.
- Accelerator specialists can easily refer to the utilities data.
- The acquired utilities data can be stored in a database for the accelerator control.
- The facility management system cannot access to the accelerator control system.
- The accelerator control system and facility management system must remain independent from each other. This is because the facility management system does not have a shutdown period. Even if the utilities DAQ system or the accelerator control system is stopped during a shutdown period for the maintenance, it will not affect the facility management system and facility operations.
- The utilities DAQ system can provide a future option to control the facility equipment from the accelerator control system without a large modification. We anticipate a future need to control the facility equipment from the accelerator control system to achieve more precise beam control for SPring-8 II [3].

Configuration

Generally, a facility management system using the BACnet protocol does not need a fast DAQ. In the facility

#ishii@spring8.or.jp

management system for SPring-8, it takes 12 minutes to update all of the analog signals. There are about 16,000 points, including analog and digital signals, in the entire facility management system. At the SPring-8 storage ring, accelerator specialists found that the preservation of thermal equilibrium in the machine tunnel and precise cooling water temperature control for the magnets and vacuum chambers contributed to beam orbit stability [4]. They also reported that one of the sources of beam kick was the expansion and contraction of the floor and base girders caused by the variation in the air temperature in the machine tunnel, and that the source of this variation was the variation in the air velocity caused by the fluctuation in the power voltage of the fan coil units [5]. The machine-cooling system must acquire precise values for the temperature, resistivity, flow, and pressure of the machine-cooling water with a high sampling rate (2 s) over a long period of observation. Data acquisition via BACnet cannot satisfy a fast DAQ.

Figure 1 shows a schematic diagram of the connection between the accelerator control system and the facility management system. The red background shows the accelerator control system with the MADOCA framework. The blue background shows the facility management system with the BACnet protocol. The green areas show the FL-net [6], and purple areas show a MELSECNET [7], which is the data link protocol for the programmable logic controllers (PLCs) of Mitsubishi Electric Co., Ltd. An iCONT is an intelligent controller connecting a local device to the BACnet backbone. The dashed line shows the gateways of the fast DAQ.

We built three types of gateways as the key devices for the utilities DAQ system. The first is a MADOCA/FL-net gateway that consists of a VME computer with an FL-net interface board for the MADOCA data acquisition software (MADOCA/FL-net GW). Second is a BACnet/FL-net gateway that consists of PLC modules with both BACnet and FL-net interfaces (BACnet/FL-net GW). The acquisition cycle from the MADOCA framework via the BACnet/FL-net GW is 60 s. Third is a MELSECNET/FL-net gateway that consists of PLC modules with both MELSECNET and FL-net interfaces for the fast DAQ (MELSECNET/FL-net GW). The acquisition cycle via the MELSECNET/FL-net GW is 2 s. An iCONT translates the data of the facility equipment data such as the electric and air conditioning utilities with a local controller to the BACnet protocol. A BACnet/FL-net GW translates the data to the FL-net protocol. A MADOCA/FL-net GW translates the data into MADOCA messages. Finally, the data are stored in the database for accelerator control. The data of the machine-cooling utilities are also stored in the database via the MELSECNET/FL-net GW and MADOCA/FL-net GW. In all, there are a BACnet/FL-net GW, four MELSECNET/FL-net GWs, and five MADOCA/FL-net GWs in the utilities DAQ system. FL-net is an Ethernet-based open standard protocol on UDP/IP for a factory floor network. We previously introduced control systems that use the FL-net such as the linac interlock system [8] in SPring-8, and the radiation monitoring system and

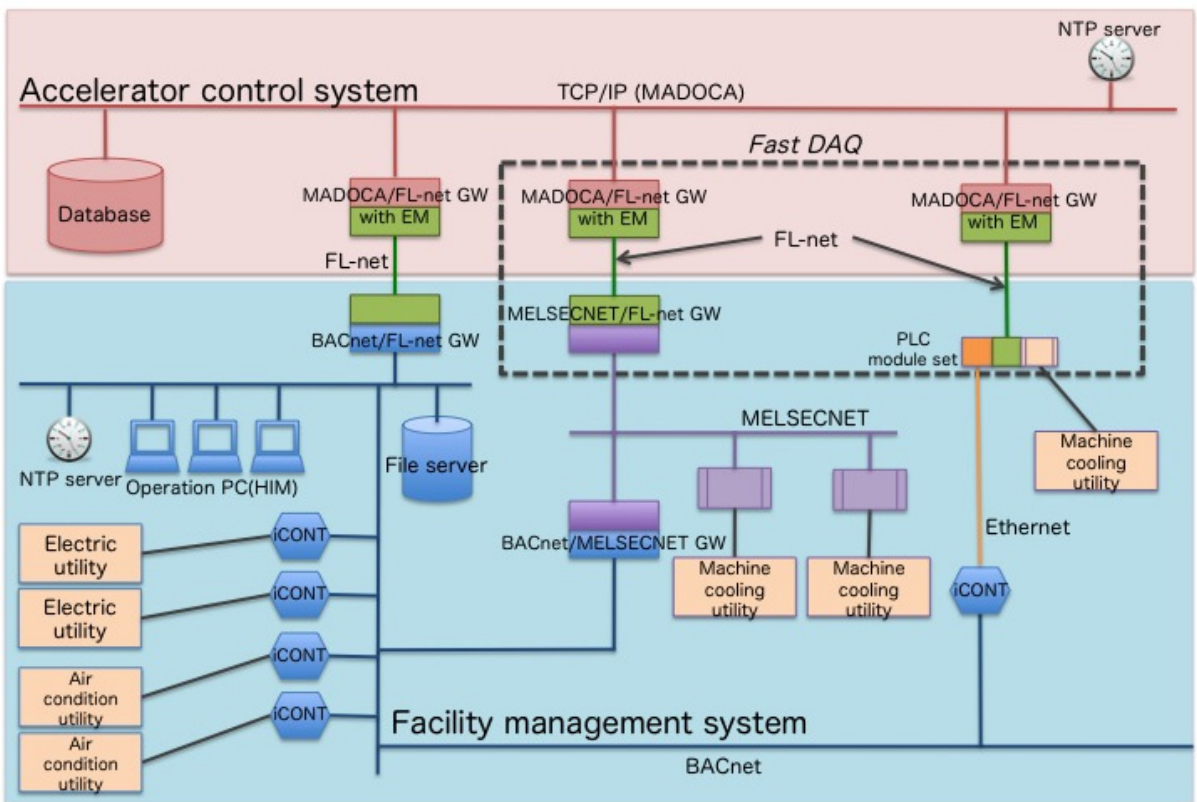


Figure 1: Schematic diagram of connection between accelerator control system and facility management system.

facility control system [9] in the XFEL facility, SACLA. These systems continue to operate stably.

A firewall is not needed for access control between the accelerator control network and facility management network when implementing the FL-net protocol. This is because a VME-based FL-net interface board implementing a MADOCA/FL-net GW can control front-end devices through the VME bus system and has a function to execute the FL-net protocol using the firmware. The FL-net board reads and writes 8 k words and 8 k bits of cyclic data transferred by the BACnet/FL-net (MELSECNET/FL-net) GW.

Future Option

The current system has been providing only data acquisition from the facility equipment. We have the option of controlling the facility equipment such as the machine-cooling utilities directly from the accelerator control system. We can achieve only minor software modification of a MADOCA/FL-net GW and a BACnet/FL-net GW. With this option, the flow of control commands would have only one path via a BACnet/FL-net GW. At the facility management system, the commands sent from the operator PCs in the central monitoring room are executed via BACnet. In the same way, the commands from the accelerator control can be sent directly to the facility equipment through a BACnet/FL-net GW with this option.

IMPLEMENTATION

We had to accomplish this large project over a short period of time. Additionally, we had limited manpower when constructing the utilities DAQ system. To solve these problems, we outsourced the construction of all the GWs to Mitsubishi Electric Building Techno Service Co., Ltd. [10] in June 2010 and Hitachi Zosen Corporation [11] in February 2011. In particular, it was a first trial of the development of Equip Management (EM) and poller, which were applications of a MADOCA framework and were run on Solaris 10 in a VME bus system. We provided a development tool that allowed MADOCA to set up configuration tables for the contractors. This development tool used OpenOffice.org [12]. It was very helpful at decreasing careless mistakes and saved work. The contractors did not need to develop a new C language function for EM and poller because they could use our prepared C libraries for FL-net device access.

As a test environment, we set up a test-database and a test-LAN using dark optical fiber cables. This test environment simulated the accelerator control system and was completely independent from any system at SPring-8. Even during the SPring-8 user time period, the contractors could confirm the capabilities of the utilities DAQ system in the test environment. Additionally, the contractors did not need to change the installation location and network configuration of all the GWs for the test environment. After the confirmation, we could smoothly install the GWs into the accelerator control system from the test environment in only one day. During this work, we frequently verified the progress.

SUMMARY

We designed and constructed a utilities DAQ system composed of three types of gateways: a VME-based MADOCA/FL-net GW, PLC-based BACnet/FL-net GW, and MELSECNET/FL-net GW between the accelerator control system and facility management system. This system can acquire the utilities data from the facility equipment with the required precision and sampling rate and can store the data in a database for the accelerator control. We also provide the future option of controlling the facility equipment from the accelerator control system. For the implementation, we outsourced the construction of the gateways, including data acquisition software of the MADOCA framework, to overcome the problems imposed by our limited manpower and short construction period. We were able to smoothly install the GWs in the accelerator control system by effectively using a test environment. The number of acquired signals is 678 signals with a slow sampling rate and 382 signals with a fast sampling rate.

Figure 2 shows the correlation between the delay of the trigger jitter at a linac M6 thyratron and the receiving voltage in July 2011. One of the sources of instability for the linac RF output is the jitter delay of a thyratron. As can be seen in figure 2, when the receiving voltage increases, the jitter delay decreases. Accelerator specialists installed an automatic voltage regulator (AVR) in the power supply of an M6 thyratron in August 2011. They are considering the installation of AVRs into other thyratrons. It is now possible to easily obtain and refer to the utilities data from the database because of the construction of the utilities DAQ system.

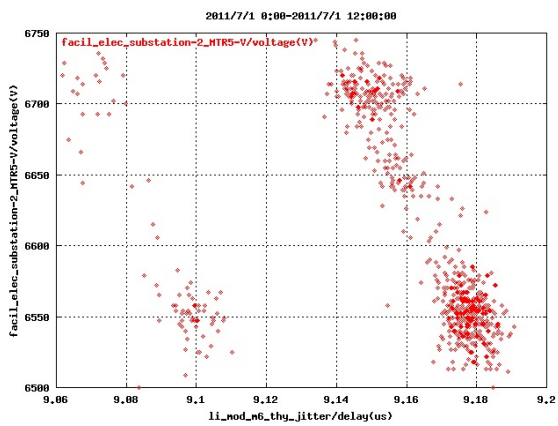


Figure 2: Correlation between delay of trigger jitter at linac M6 thyratron and receiving voltage.

REFERENCES

[1] R. Tanaka et al., "The first operation of control system at the SPring-8 storage ring", Proc. of ICALEPCS'97, Beijing, China, 1997, p 1.
[2] http://www.bacnet.org
[3] T. Watanabe et al., "Current Status of SPring-8 Upgrade Plan", Proc. of IPAC 2011, San Sebastian, Spain, 2011
[4] H. Tanaka et al., "Beam orbit stabilization at the SPring-8 storage ring", Proc. of IWAA2002, SPring-8, Japan, 2002, p 1.
[5] S. Matsui et al., "Daily fluctuation of electron beam due to electric power voltage of air conditioner in the storage ring tunnel for synchrotron radiation", Proc of Particle Accelerator Society of Japan, Sendai, Japan, 2006, p 245. (in Japanese)
[6] http://www.jema-net.or.jp/Japanese/standard/opcn/opcn102.html
[7] http://wwwf2.mitsubishielectric.co.jp/melfansweb/english/index.html
[8] T. Fukui et al., "Development of a communication with PLC by using the FL-net as an open standard PLC link", Proc of PCaPAC2005, Hayama, Japan, 2005
[9] T. Masuda et al., "Facility Utility Control System of XFEL/SPring-8", Proc of ICALEPCS2009, Kobe, Japan, 2009, p 286.
[10] http://www.meltec.co.jp
[11] http://www.hitachizosen.co/jp/english/index.html
[12] http://www.openoffice.org/

THE CONTROL AND DATA ACQUISITION SYSTEM OF THE NEUTRON INSTRUMENT BIODIFF

Harald Kleines, Matthias Drochner, Lydia Fleischhauer-Fuss, Frank Suxdorf, Michael Wagener, Stefan van Waasen, ZEL, Forschungszentrum Jülich, 52425 Jülich, Germany
Tobias Schrader, JCNS, Forschungszentrum Jülich, 52425 Jülich, Germany
Andreas Ostermann, FRM-II, Technische Universität München, 85747 Garching, Germany

Abstract

The Neutron instrument BIODIFF is a single crystal diffractometer for biological macromolecules that has been built in a cooperation of Forschungszentrum Juelich and the Technical University of Munich. It is located at the research reactor FRM-II in Garching, Germany, and is in its commissioning phase now. The control and data acquisition system of BIODIFF is based on the so-called "Juelich-Munich Standard", a set of standards and technologies commonly accepted at FRM-II, which is based on the TACO control system developed by the ESRF. In future, it is intended to introduce TANGO at the FRM-II. The Neutron Image Plate detector system of BIODIFF is already equipped with a TANGO subsystem that was integrated into the overall TACO instrument control system.

INTRODUCTION

In protein crystallography X-ray diffraction is a successful tool for structure analysis. Neutron diffraction is a complementary method, since it can provide detailed information on hydrogen atom locations [1]. A further advantage is the possibility to differentiate between hydrogen and deuterium atoms. The development of Neutron Image Plate (NIP) detectors was a big advance for neutron protein crystallography leading to the developments of several instruments, e.g. the BIX-3 at the Japan Atomic Energy Research Institute and LADI-III at the Institute Laue-Langevin [2]. Also at the German research reactor FRM-II in Garching a single crystal diffractometer for biological macromolecules called BIODIFF has been developed as a joint project of Forschungszentrum Jülich and Technische Universität München (TUM). BIODIFF is a monochromatic instrument using a cylindrical NIP as its main detector. For adjustment purposes it is equipped with a CCD camera looking at a ZnS:Li scintillator as an auxiliary detector [3]. The complex mechanics of BIODIFF includes about 20 movement axes, mainly equipped with stepper motors. Additional control tasks are related to beam shutter control and sample environment. The control and data acquisition system of BIODIFF is based on the "Juelich-Munich Standard". This is a joint effort of ZEL (Central Institute for Electronics of Forschungszentrum Jülich) and TUM to define a common framework for the electronics and software of neutron instruments that is followed by most instruments at the FRM-II [4]. It is based on the TACO control system developed by the ESRF and the extensive use of industrial type front-end equipment, e.g. PLCs, fieldbus systems or remote I/Os.

OVERVIEW OF THE BIODIFF INSTRUMENT AT FRM-II

According to Fig. 1 the BIODIFF consists of two subsystems, the monochromator with its outer shielding and the detector housing.

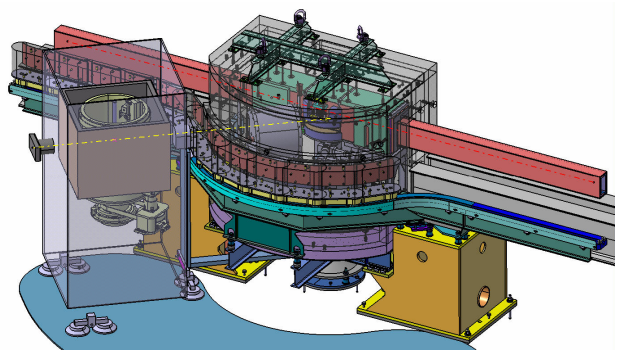


Figure 1: Schematic view of the BIODIFF instrument.

The incoming neutron beam is monochromated by a graphite crystal, which is movable in 6 degrees of freedom. An additional velocity selector from ASTRIUM removes the higher order wavelength contaminations of the monochromatic beam. By movement of graphite crystal, selector and detector housing a neutron wavelength between 2.4 Å and 5.6 Å can be selected. For the interruption of neutron beam and γ-radiation two shutters are foreseen.

The detector housing contains the cylindrical NIP as well as the CCD camera (icon-L series from company ANDOR), which both can be moved in and out by several mechanical axes. The NIP detector system is commercially available from company "Maatel". It contains the PMT-based reading head as well as the erasing system. Control of NIP drum movement, erasing and readout is implemented by Maatel as a self-contained system including an operating PC. On this PC a complete TANGO system is implemented, providing interactive local control as well as remote control via the CORBA-based TANGO protocol.

The NIP detector provides a solid angle of almost 2π, with the sample in the centre. Sample environments can be mounted on the sample table above the NIP detector, which is movable in three degrees of freedom. Additional fine positioning of the sample can be done with piezo-driven manipulators from company "attocube systems".

THE "JUELICH-MUNICH STANDARD"

The "Jülich-Munich standard" is a framework for the selection of technologies and components at each level of the control system. The definition of this framework was motivated by synergy effects and the reduction of spare parts on the shelf. A guiding principle for the framework was to minimize the development efforts and to acquire as much from the market as possible. A key component of the framework is the consistent use of industrial technologies like PLCs, fieldbus systems or decentral periphery in the front end. Main motivations are:

- low prices induced by mass market,
- inherent robustness
- long term availability and support from manufacturer
- powerful development tools

A control system according to the Jülich-Munich Standard is organized hierarchically into the following levels:

Field level: The field level is the lowest level, at which devices that are not freely programmable reside, like motor controllers, SSI controllers, PID controllers, analogue and digital I/O modules, or measurement equipment. For all industrial type of I/O modules PROFIBUS DP based decentral periphery is recommended. Siemens ET200S is the preferred one. Jülich Centre of Neutron Science (JCNS, Neutron science institute of Forschungszentrum Jülich with outstation at FRM-II) predominantly uses the stepper motor controller 1STEP from Siemens.

Control level: The control level resides on top of the process level. Devices at the control level are freely programmable. They must meet real time requirements and guarantee robust operation in a harsh environment. At the control level Siemens S7 PLCs, mainly from the S7-300 family, are used, because they dominate the European market.

Process communication: Process communication covers the communication of devices at the field and control level with supervisory controllers or computers. For lab equipment GPIB and proprietary RS232/RS485 connections are unavoidable. For industrial automation equipment PROFIBUS DP is the recommended choice. It is the dominating fieldbus in Europe and is naturally supported by S7 PLCs and many other devices. A major reason for its success is the technological and functional scalability based on a common core as well as the programming model, which easily maps to PLC operation.

Experiment Computer: For economical reasons, all experiment computers should be PCs. Linux, being well established in the scientific community, is the only supported operating system. There is no definition of a specific kernel version or distribution. Direct device access should not be implemented on conventional PCs but on CompactPCI systems. CompactPCI allows deploying a variety of existing software in a mechanically more robust platform that fits into 19" racks.

Middleware: Since the framework aims at an inherently distributed system, software support for the transparent distribution of services between systems is required. For this purpose TACO has been selected as the middleware system. TACO is a client-server framework developed for beam line control at the ESRF in Grenoble. In a TACO environment each device or hardware module is controlled by a TACO server. The server offers a set of device-specific functions, which can be accessed by TACO clients via a RPC-based mechanism over a TCP/IP network. To make its functions available to clients, the device server registers itself with the so called "manager" process. The manager operates as a name server, which is consulted by clients to get the actual location of a server. TACO includes a simple database for sharing of configuration data and operational variables between clients and servers.

Application level: On the client side, two variants of application programs are used: Where flexibility is desired and no GUI is needed, the scripting language Python is used. More static GUI applications are implemented in C++, using the "Qt" class library, with TACO access provided by device specific C++ wrapper classes. Recently, due to pressure from instrument scientists a macro language consisting of fixed command words and their corresponding arguments was defined. The idea was to have an extremely simple tool for standard scans without having to deal with the complexity or traps of a full programming language. As a consequence, the macro language has no control structures or variables and instead of loops it provides a "scan" command specific for each instrument. An interpreter for this command language was first implemented in python using the python "cmd" module on the neutron reflectometer MARIA [5].

THE BIODIFF CONTROL AND DATA ACQUISITION SYSTEM

Physical Architecture of the Control System

According to Fig. 2 the control and DAQ system is implemented as a distributed system with a hierarchical architecture. On top of the system resides the so-called control computer with all application software – GUI-based as well as script-based. Via the experiment network the control computer accesses the "server computers", to which all front end systems (detectors, monitors, position encoders, motor controllers, …) are attached. On the "server computers" TACO servers are running, which access the peripheral devices via dedicated device drivers. The only exception is the computer provided by Maatel for the NIP system, where a TANGO server is running.

The "slow control" peripherals are indirectly connected to the "server computers" via a PROFIBUS segment with the main S7-300 PLC. Stepper motor controllers and SSI modules as well as digital and analogue I/Os reside in ET200 decentral periphery systems, which are connected to the PLC via an additional subordinate PROFIBUS

segment. The safety PLC (ET200S F-CPU), the adapter modules for encoders with NDAT interface as well as a touch panel for local operation of the PLC are connected to the same subordinate PROFIBUS segment.

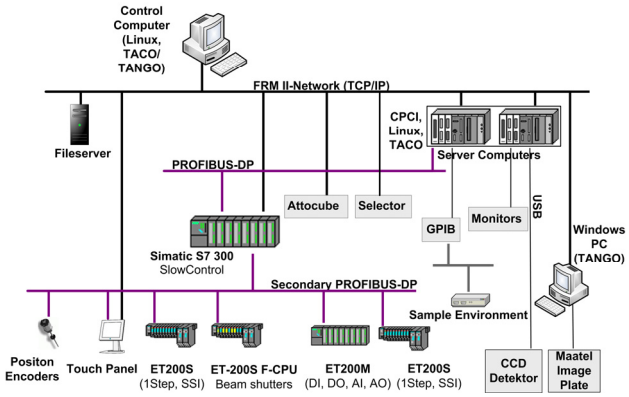


Figure 2: Physical architecture of the BIODIFF control and data acquisition system.

The CCD camera is connected to a server computer via USB. Selector and the controller for the attocube positioners are connected to a server computer via Ethernet.

Software Architecture

As shown in Fig. 3, the implemented software is distributed between three levels of the system hierarchy. All software below the lower dashed line runs on PLCs in the front end. The software modules shown between the dashed lines are running on the server computers. This comprises TACO servers and device drivers for dedicated HW modules, e.g. detector electronics, counter/timer board, PROFIBUS controller or GPIB controller. The TACO middleware is the glue that connects the server computers to the control computer, where the client application programs as well as the TACO manager and database (all above the upper dashed line) are running. Since TACO is location-transparent, the application programs could run on any Linux-based system.

A thin abstraction-layer implemented in Phython above the generic TACO-Python binding hides many details from the user, e.g. it allows the use of symbolic names and provides the conversion between device units and physical units. This abstraction layer provides a comfortable script access to all spectrometer features. On top of this abstraction layer the above-mentioned macro language has been implemented, which is the standard tool to do measurements with the Instrument. Measurement data are stored in the NeXus format. It is intended to implement a GUI-based measurement program in future, but its functionality has not been defined yet. Up to now only one GUI-based program has been implemented, which visualizes the mechanical setup of the instrument and allows interactive control of each axis, in order to support service by technical personnel.

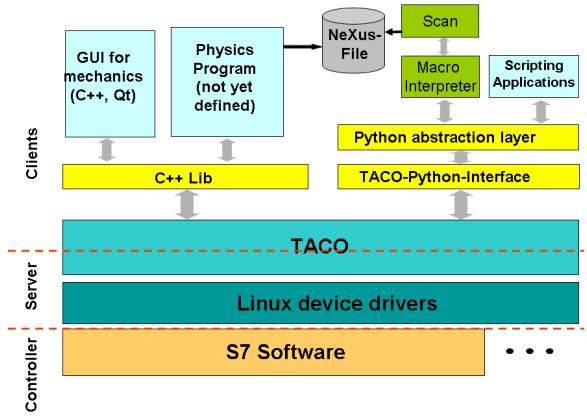


Figure 3: BIODIFF Software structure.

CONCLUSION AND OUTLOOK

Since the "Juelich-Munich Standard" is the common, powerful toolbox used in all JCNS instruments, the implementation of the control and data acquisition system was straight-forward. Especially the integration of the TANGO subsystem controlling the NIP was without any problems, due to the generic TANGO/python binding and because of the similarities between TACO and TANGO. The BIODIFF started commissioning in October 2010, when it saw first neutrons. Due to a one year shutdown of FRM-II the commissioning had to be interrupted and is expected to continue in October 2011.

Currently the introduction of TANGO for the neutron instruments at FRM-II is discussed, since TANGO is the natural successor of TACO. As a consequence, in future the BIODIFF control and data acquisition system may switch from TACO to TANGO

REFERENCES

[1] B.P.Schoenborn, "Neutron protein crystallography", TIBS, September 1977, pp. 206

[2] N. Nimura et. al., "High resolution neutron protein crystallography.", Z. Kristallogr. 218 (2003), September 1977, pp. 96

[3] M. Monkenbusch et. al., "BIODIFF: Single crystal diffractometer for biological macromolecules", JCNS Experimental reports 2007/2008, pp 58, http://www.jcns.info.

[4] H. Kleines et al., "Implementation of the Control and Data Acquisition Systems for Neutron Scattering Experiments at the New "Jülich Center for Neutron Science" According to the "Jülich-Munich Standard", Proceedings of the ICALEPCS 2005, Geneva, 2005.

[5] M. Drochner et al., "New Developments for Neutron Scattering Insturments", Proceedings of the ICALEPCS 2009, Kobe, 2009.

CONTROL SYSTEM FOR MAGNET POWER SUPPLIES FOR NOVOSIBIRSK FREE ELECTRON LASER

Yu.M.Velikanov, V.F.Veremeenko, N.A.Vinokurov, A.A. Galt, B.A.Dovzhenko, V.R.Kozak, E.A.Kuper, L.E.Medvedev, A.S.Medvedko, S.S.Serednyakov.

BINP SB RAS, Novosibirsk, Russia

Abstract

The control system for the magnetic system of the free electron laser (FEL) is described. The characteristics and structure of the power supply system are presented. The power supply control system based on embedded intelligent controllers with the CAN-BUS interface is considered in detail. The control software structure and capabilities are described. Besides, the software tools for power supply diagnostics are described.

INTRODUCTION

A high-power free electron laser based on the accelerator-recuperator [1] is under construction now at Budker Institute of Nuclear Physics. The first and second phases of the project were commissioned recently. As with other particle accelerator facilities, the magnetic system of the FEL is an important part of the installation. It consists of many magnetic elements of different types: bending magnets, quadrupole lenses, and correctors. In total, the first and second stages contain about 80 and about 120 magnetic elements, correspondingly. The windings of all elements are fed from DC current power supplies designed and manufactured at BINP. These power supplies are in turn controlled by analog-to-digital and digital-to-analog converters (DACs and ADCs), which are connected to the control computer with the CAN-BUS interface [2]. The control system of the power supplies of the magnetic elements is described in this article.

POWER SUPPLY SYSTEM

As mentioned before, the magnetic system of the FEL consists of several types of magnetic elements. The elements differ in their influence on the electron beam and the amount of consumed DC current, which results in the use of power supplies of various types. The main characteristics of the power supplies applied are shown in Table 1. This table also presents the quantity of power supplies of different types for the first and second stages of the FEL. All other characteristics of the power supplies are presented in detail in [3]. One can see from Table 1 that the amount of low-power (3A and 10A) supplies used is large, while the number of high-power (300A, 1000A, and 2500A) supplies is very small. In addition, the low-power supplies are very small and do not require water cooling. Therefore it is very convenient to combine those numerous low-power supplies in one rack, their control joined in one multi-channel control device. At the same time, high-power supplies require a larger space and more parameters to measure and control, and hence an individual control device.

Table 1: Main Parameters of the Power Supplies

Imax, A	U max, V	Qty		Magnetic elements
		1st stage	2nd stage	
±3.0	12	90	125	Correctors
±10.0	20	54	70	Bending magnets, Quadrupole lenses
300	12	0	5	Bending magnets, Quadrupole lenses
1000	12	2	2	Bending magnets
2500	48	1	1	Undulator

CONTROL HARDWARE

As mentioned above, all control devices used in this system have the CAN-BUS interface and are connected to one CAN-line. Using the CAN standard for this system allows integrating the control of different power supplies, sometimes of different volume and considerably spaced. Depending on the details of this system operation, required accuracies, dimensions and consumed power, all power supplies could be divided into groups according to the method of their connection to the control hardware:

1) low-power supplies operated by multi-channel control devices and
2) high-power power supplies operated by precision single-channel control devices

For the first group, a **CANDAC16-CANADC40** pair is used. **CANDAC16** is a 16-channel digital-to-analog converter, and **CANADC40** is a 40-channel analog-to-digital converter. This pair is used to control a set of 16 power supplies. **CDAC20** devices are used for control of power supplies of the second group, which have higher accuracy in comparison with those of the first group, but one device can control only one power supply.

Table 2: Parameters of the Control Devices.

Parameter	Candac16	Canadc40	Cdac20
DAC resolution	16 bits	-	21 bits
DAC accuracy	0.05%	-	0.005%
DAC channels	16	-	1
ADC resolution	-	23 bits	23 bits
ADC accuracy	-	0.03%	0.003%
ADC channels	-	40	5

Scale (input/output)	10 V	10 V	10 V
Qty of devices used in the 1st stage	10	10	3
Qty of devices used in the 2nd stage	24	24	7
Qty of devices used in the 3rd stage	24	24	12

All the above control devices are developed and manufactured at BINP [4]. Table 2 presents the main characteristics of all the devices as well as the quantity of devices applied in the 1st, 2nd and 3rd stages of the FEL.

Figure 1 shows the arrangement of the power supplies and control devices and their connection with the CAN-BUS.

The low-power supplies are grouped in sections, 16 power supplies in one section. Each rack contains up to 3 sections. Each section is controlled by a **CANDAC16-CANADC40** pair.

The way of connection of high-power supplies to their controllers depends on the size of the power supply and presence of additional electronics.

All above controllers are connected to a single CAN line, which is connected to the control IBM PC. It should also be noted that all control devices can operate both in the single-channel mode and the multi-channel one. This ability allows one to organize single-channel measurement of the output current of power supply "on demand", without interrupting the usual measurement cycle for the output currents of other power supplies.

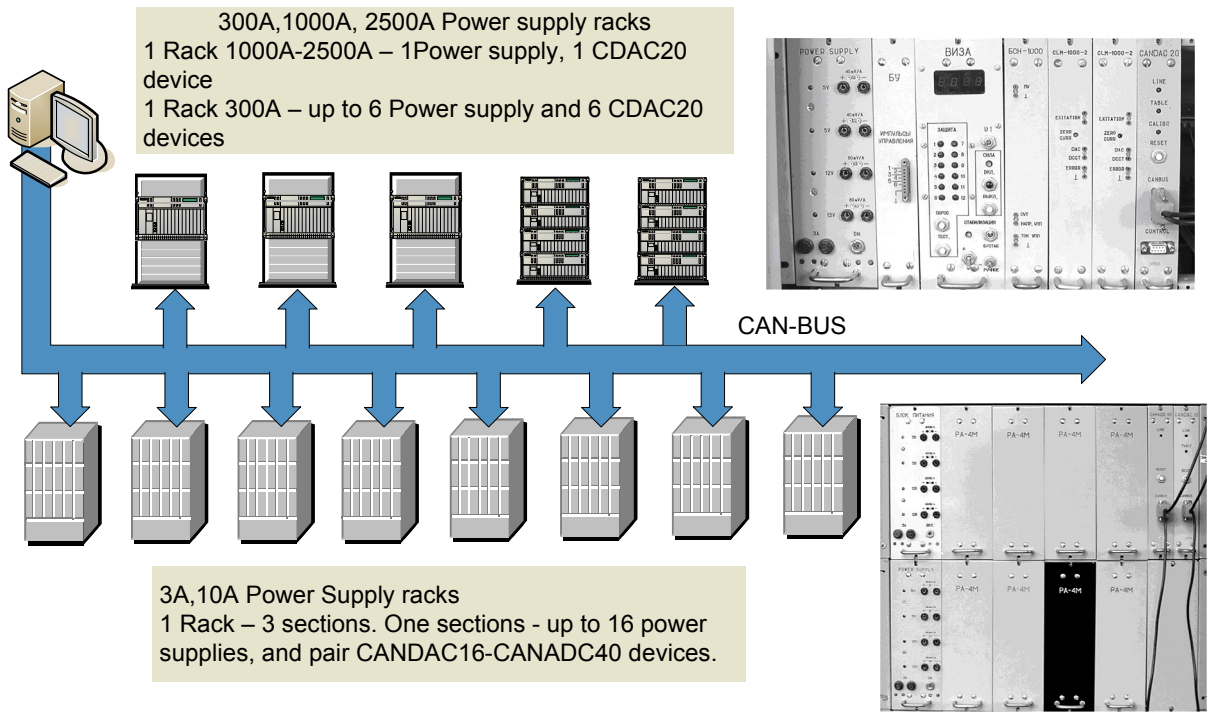


Figure 1: Arrangement of the power supplies and their connection to the CAN-BUS.

CONTROL SOFTWARE

The software controlling the operation of all the power supplies is implemented in one application. The main features of the program are as follows:

1) an advanced user interface, allowing full control of the power supply operation;

2) remote control of all the power supplies using the Epics Channel Access Server, i.e., one can set and check the currents of the magnetic elements from any other computer in this local network;

3) demagnetization of all magnetic elements of the system via applying current of an alternate polarity to an element and amplitude damping from the maximum source current to zero;

4) the possibility of saving/loading the currents of all or an arbitrary set of the power supplies;

5) a set of tools for diagnostics of the power supply operation: real-time verification of the output current and measurement of ripples in an automatic mode;

6) similar operation in all FEL configurations: the 1st, 2nd and 3rd stages.

The main window of the control software is presented in Fig.2. The power supplies and their states are shown in lines, containing the name of a magnetic element fed by this power supply and its current at this moment. The color of line and its content reflects the actual state of the power supply.

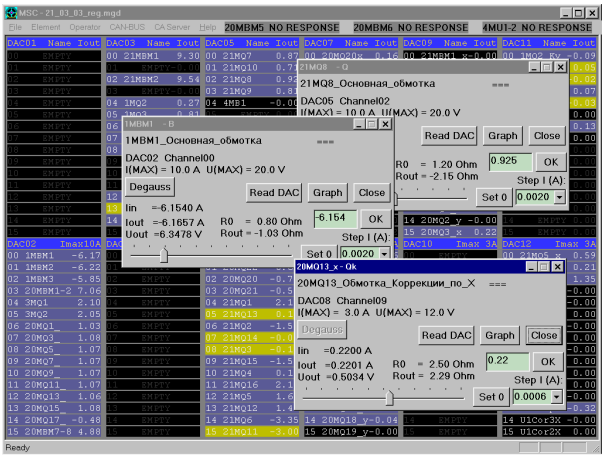


Figure 2: Main window of the control software.

Since the number of magnetic elements is very large, it is very difficult to find a particular element in this list. There is a mnemonic scheme of the entire FEL facility with magnetic elements that simplifies search for a required element. The scheme is shown in Fig.3.

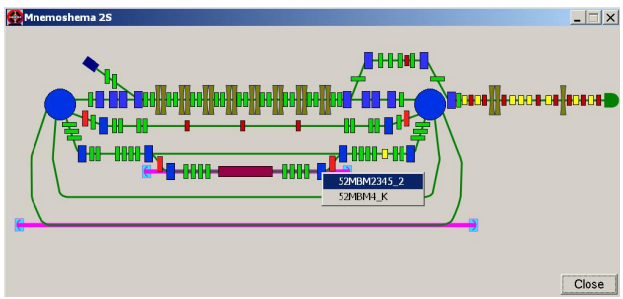


Figure 3: Mnemonic scheme of the FEL magnetic system.

To change current in a required element, the user has to open an individual dialog window, clicking on the corresponding element in the mnemonic scheme or the corresponding colored line in the main window.

As mentioned above, the control application also has some tools for diagnostics of the regularity of the power supply operation. One of them is the real-time comparison of current set for a power supply and a measured current value. If the difference exceeds a certain limit, the line in the main window that corresponds to this power supply is marked yellow (warning) or red (alarm). The program also watches the time stability of current set for a power supply. If the current changes in time by a value exceeding a certain limit, this power supply will be marked with color too.

Another diagnostic tool for the power supplies is a the control of ripples of output current with frequencies of up to 500 Hz. To implement this, the test program runs a cycle of single-channel measurements on all ADC channels, measuring the output currents of all the power supplies. Processing of one power supply includes construction of a measurement array of a required length and mathematical treatment of the data, i.e. finding the average current, maximum deviation and root-mean-square deviation of the current. Results are displayed in a separate window, shown in Fig. 4. The values of maximum deviation and root-mean-square deviation of current are presented graphically with vertical bars. The entire array of measurements is also displayed. The ripples, if any, can be detected by the magnitude of the color bars and analyzed in the graph of the array of measured values.

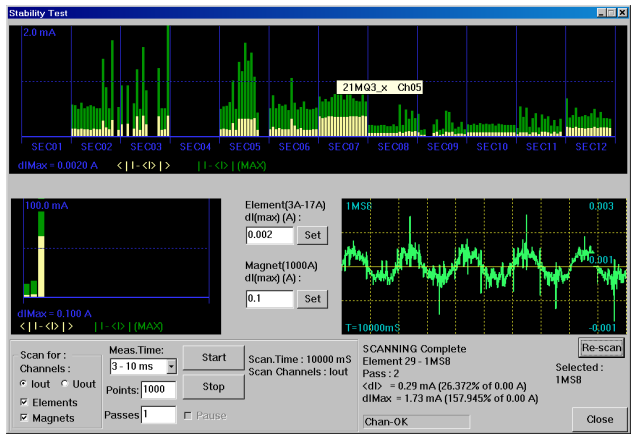


Figure 4: Window of the test of the power supply ripples.

In addition, for the purpose of remote control of the power supplies, the EPICS portable channel access server is built in the control application. Each power supply is presented with 3 process variables (PVs) on this server: a PV for the specified current, a PV for the measured current, and a PV for the measured voltage of the power supply.

CONCLUSION

The control system of the magnet power supplies has been operating for 8 years, demonstrating high stability and convenience of use. The diagnostic tools allow full control of the regularity of the power supply operation. The built-in EPICS channel access server allows integrating the control software to the EPICS-based control system.

REFERENCES

[1] N.A. Vinokurov et al. Novosibirsk free electron laser facility: Two-orbit ERL with two FELs. Proceedings of FEL2009, TUOD01

[2] http://www.can-cia.org CAN In Automation. 2009.

[3] Yu.M.Velikanov et al. Free electron laser for Siberian centre for photochemical research: The control system for the magnet power supplies. Proceedings of RuPAC XIX, WEHP18

[4] V.R.Kozak, E.A.Kuper. Microprocessor controllers for power sources. BINP preprint 2001-70, 2001.

THE MRF TIMING SYSTEM. THE COMPLETE CONTROL SOFTWARE INTEGRATION IN TANGO

J. Moldes, D. Beltrán, D. Fernández, J. Jamroz, J. Klora, O. Matilla, R. Suñé
CELLS, Cerdanyola del Vallès, Barcelona, Spain

Abstract

Alba [1] is a synchrotron light source under installation located nearby Barcelona. This 3 GeV third generation light source is planned to deliver the first X-rays beam to the users in 2012. The linac was commissioned in 2008, the booster in 2010/2011 and the Storage Ring in 2011. The seven beamlines included in the "Phase One" are being commissioned at the end of 2011.

The timing system is mainly used to synchronously distribute signals to all the equipments of the machine that need them. It is built on top of the MRF [2] timing solution. It is a fast digital, point to point, event based system. The system basically works by sending event codes from one event generation source (synchronized with RF/4 frequency) through a fast, tree structured, bidirectional fibre optics cabling network to many event receivers (about 100), which react to these event codes in different ways, amongst which the most commonly used is to generate a pulse (which characteristics are configurable) in one of its outputs. This pulse is connected to the hardware equipment that physically needs it. There are currently 418 output signals connected to different equipments. In other words, timing provides the "synch" in the synchrotron, keeping synchronized all the key systems (power supplies, RF and diagnostics).

The receivers are also capable of sending event codes up to the event generator when they detect activity in one of their two inputs. The generator can then redistribute this event again in downwards direction to all the receivers. This feature has been used to implement the Fast Interlock system [3]. There are currently 49 interlock signals connected to the system.

Logging of events with globally distributed timestamps is available in the receivers. This feature is specially useful (coordinated with Fast Interlock system) to easily and quickly determine the cause of a sudden beam loss.

The event generator is also capable of distributing up to 8 arbitrary digital signals using another feature called Distributed Bus (DBus).

In combination with other systems, timing will also be the key for getting the desired filling pattern of the machine and for implementing the top-up feature.

LAYOUT

The layout of the system is detailed in Fig. 1. As pointed above, it is basically a tree structured fibre optics network, starting from the event generator (EVG from now on) fanned out by the fanout/concentrators (fanouts from now on) and ending in the event receivers (EVR[s] from now on). This downwards path is the one mainly used. The EVG sends event codes to the EVRs, which react to them mainly by generating a pulse in any of its outputs.

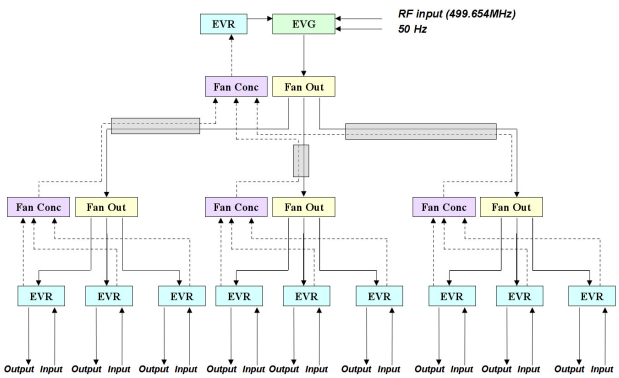


Figure 1: Layout of the Timing System.

However, the communication can also be done in upwards direction, starting from any of the EVRs sending an event up to the EVG. The later can then propagate downwards the event to all the EVRs. This is the core of the Fast Interlock system [3].

THE HARDWARE

Hardware was provided by different companies.

The core of the system, formed by the EVG, EVRs and fanouts, was supplied by MRF [2]. The EVG is the 230 model and almost all the EVRs are also 230 model. In addition to these, a few units of EVRTG300 have been acquired for special purposes, which we will see later.

It was decided to host the EVG and EVRs in cPCI machines, due to their robustness and reliability. Hence, the EVG and EVRs (model 230) were supplied in cPCI 3U form. The 3U cPCI chassis cPCIS-2632 and CPUs cPCI-3840 were supplied by Adlink [4].

The fanouts come in 6U cPCI form. Adlink chassis cPCIS-6418U and cPCIS-6130R were selected to host these cards. Fanouts control is much simpler than EVG and EVRs and hence it is not necessary to use a CPU to manage them. They provide an ethernet connection and an API based on UDP that is enough to control them.

The EVRTG300 units are also 6U cPCI form and were mounted in cPCI-6130R chassis with a cPCI-6965 CPU.

The fibre optics cables were supplied by R&M [5].

In the next section we will have a more detailed view of the main hardware components (EVG, EVRs and fanouts) and their main capabilities. Thought extensive, it is not a

complete description of these devices. See [2] for full information.

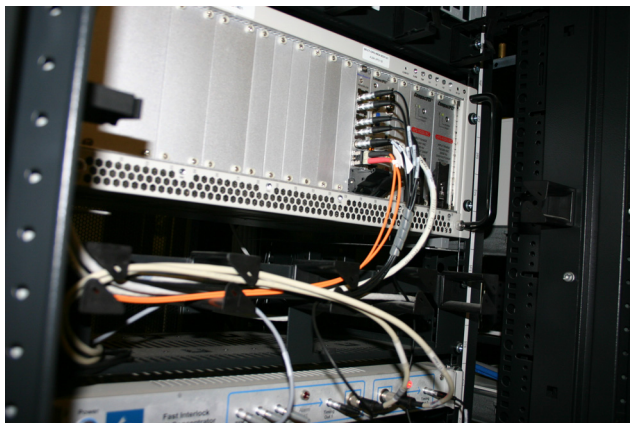


Figure 2: cPCI chassis with CPU and 1 EVR.

Figure 2 shows an example of use of the system. We can see a cPCI chassis with a CPU and an EVR. The four outputs of this EVR are used in this case to provide timing to 8 BPM units. We can also see two inputs coming from a Fast Interlock concentrator [3], in this case coming from any of the 8 BPM units in case the beam orbit gets out of limits. The two orange Rx/Tx fibre optics cables are also visible in the figure.

Event Generator

The EVG is responsible for creating and sending out the events to the EVRs.

Events are sent out by the EVG as event frames (words), which consist of an eight bit event code and an eight bit distributed bus data byte. The maximum event transfer rate is derived from the external RF/4 clock. The optical event stream transmitted by the EVG is phase locked to the clock reference.

There are several ways of generating the events:

- Trigger events. A single event is fired by an internal configurable counter.
- Sequence events. This is the feature used in our case. The sequence is a table like structure, with pairs timestamp - event, so the users can define exactly the time to wait until the next event is sent. The sequence itself is triggered by an internal configurable counter. Receivers can be grouped by subsystem to react only to some events. This way, users can fire different events that trigger only a given subsystem, without affecting the rest.
- software events: any event can manually be sent on user request.
- events received from the upstream EVRs (used for the Fast Interlock System).

In addition to events the EVG enables the distribution of eight simultaneous signals sampled with the event clock rate. This feature is called distributed bus. Distributed bus signals may be provided externally or generated on-board by programmable counters.

Event Receivers

EVRs decode timing events and signals from the optical event stream transmitted by the EVG.

The EVRs lock to the phase event clock of the EVG and are thus phase locked to the RF reference.

The EVRs basically convert the data got from the EVG to hardware outputs. Every info packet got from the EVG is 16 bits long, and is divided into two parts:

- Event code (8 bits).
- DBus bits (8 bits). Any output of the receiver can be configured to follow any of these bits. When the value of this bit is 1, the output is set to high, and it is set to low when the value is 0. This means that we can reproduce up to 8 arbitrary digital signals in any part of the machine at clock frequency resolution.

Two mapping RAMs (only one can be active at a time) are provided in order to configure how the EVR will react when receiving a given event. This RAM contains a table like structure in which event codes are associated to different actions, amongst which the most common ones are:

- Trigger a pulse in a given output. The delay since the event is got is configurable, as well as the width. Both width/delay can be set in 8 nano seconds steps.
- Set a given output to high. A delay is configurable.
- Reset a given output to low. A delay is configurable.
- Log the event. Events can be stored with globally distributed timestamps into a log memory.
- Forward the event downstream.

Two external hardware inputs are provided to be able to take an external pulse to generate an internal event. This event can also be transmitted in upwards direction to the the master EVR and from this to the EVG for distributing it downwards to all the EVRs. This is the core of the Fast Interlock system [3]: any interlock signal connected to any receiver can be distributed to the whole machine in 4.2 micro seconds.

EVRTG300 units add to the above a special feature. A fine grain delay of 10 pico seconds steps can be set in any of its outputs. This will be used for being able to inject any bucket of electrons in any position of the storage ring. It will also be used for the kicker magnets delay fine tuning and for providing timing to the streak camera.

Fanouts

These devices are mainly hardware devices and have few configurable settings. They have a dual function:

- The 8-way fan-out receives the optical event signal through a fibre connected to the uplink RX port. This signal is fanned out to all eight downstream ports.
- The concentrator receives signals from up to eight EVRs or downstream concentrators and forwards the signals upstream.

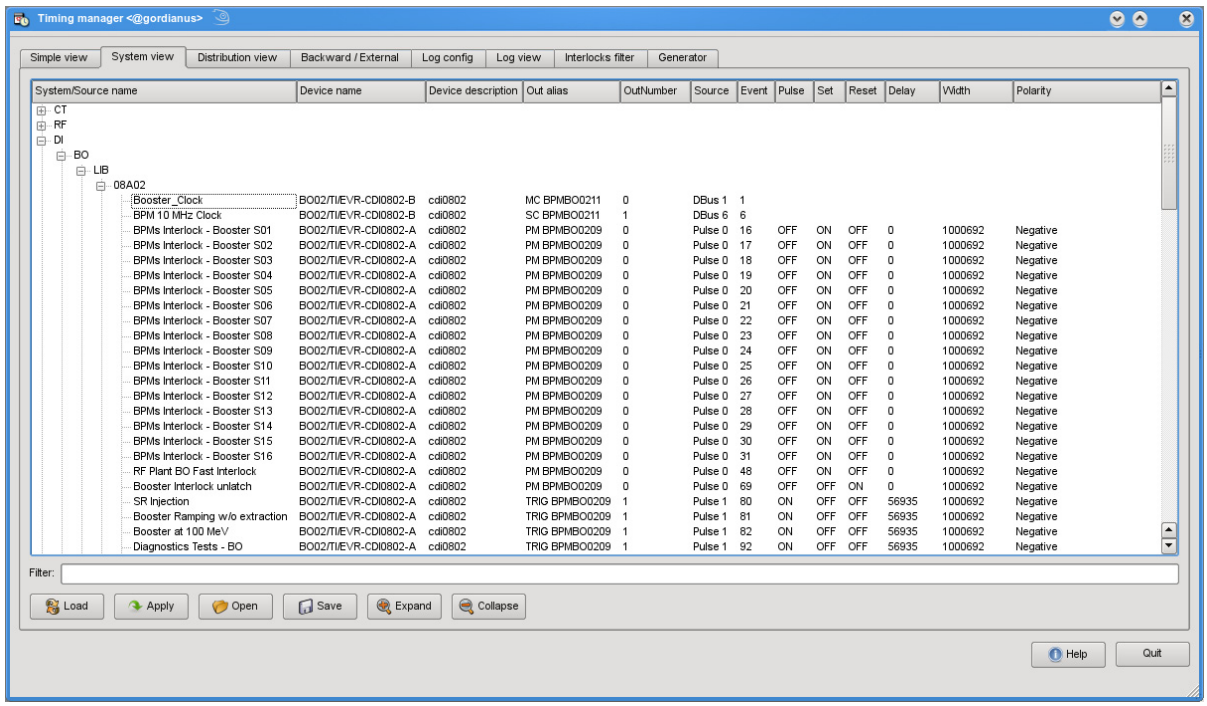


Figure 3: Timing manager GUI, the main users interface with timing system.

THE SOFTWARE

The software for controlling the system was developed from the lowest level up to the highest user GUIs level. In the middle we had to develop the core of the system: the tango device servers in order to integrate all the components into tango [6].

Drivers and API

Linux device drivers for the EVG and the EVRs had to be developed. Drivers for kernel 2.6 were developed by R. Suñé and J. Pietarinen from MRF [2].

In addition to this, a binding for python has been developed in order to use the manufacturer's API, which is written in C++. Swig [7] has been used for this purpose.

Tango Device Servers for EVG, EVR and Fanouts

Tango device servers have been developed to control all the features of the EVG, EVRs (both 230 and 300 models) and fanouts. These device servers have been written in python, using the above mentioned *pythonized* API and the python binding for Tango: PyTango [6]. They allow to control all the features of a single unit. In addition to this, the tango device server allows to permanently store in Tango database the configuration of a given unit and reload it when the device server is started. This permits the permanent storage of the configuration of the whole timing system.

Tango Device Servers for Configuring and Monitoring the System

In order to be able to control all the units in a centralized and convenient way, the TimingManager device server was developed. This device allows to set/retrieve the configuration of all the EVRs using a simple XML format and also configure/retrieve the logs.

A TimingMonitor device server was developed for monitoring the system health. This device is continuously monitoring the status of the EVG, EVRs and fanouts to find any problem that may occur.

THE GRAPHICAL USER INTERFACES

EVG and EVR Expert GUI

Expert GUIs for the EVG and EVRs (both models included) were developed. These GUIs communicate respectively with the EVG and EVR tango device servers in order to set/retrieve all the settings of the unit. These GUIs are meant to be used by expert users only.

Timing Manager GUI

This GUI is the users main interaction point with the system. Fig. 3 shows this application. This GUI runs of top the TimingManager device server.

Users provided the initial configuration of the system in a spreadsheet, basically specifying for each output of each receiver which event or events would trigger that output and also the configuration of the pulse/set/reset to be triggered. For each receiver they also provided a tag, which was used for splitting it and building the tree structure shown in Fig. 3, hence allowing users to group EVRs by subsystem and easily locate any output.

The first three tabs, show basically the same information but sorted in different ways. As shown in the figure, every line shows the output information, formed by

the tree node (including event name), the EVR tango device name, a user editable description for that EVR, a user editable alias of the output, the output number, the source for the output (DBus number or event code), pulse/set/reset (exclusive), pulse delay and width (the later no applicable for set/reset) and polarity. Delay, width and polarity are user editable.

The difference between "Simple view" and "System view" is that the former shows only one line per output (different events can trigger the same output), while the later displays the same line as many times as events trigger it. "Distribution view" show the same information as the "System view", but in this case the tree is built from the event code down to the EVRs, hence allowing users to know which outputs of which EVRs are triggered by a given event.

"Backward/External" allows the configuration of the backward events, used by the Fast Interlock system [3].

"Log config/Log view" allows the configuration of the logging of events in any EVR and retrieving/inspecting the contents of the logs of all the EVRs.

"Interlocks filter" allows to enable/disable forwarding of a given event in the main top EVR of the system (the one on the top in Fig. 1). This is useful for preventing a given interlock to be propagated downstream.

"Generator" tab allows users to enable/disable the EVG sequencer and editing it's contents.

The open/save buttons allow users to load/save the configuration of the whole timing system.

Timing Monitor GUI

This GUI is built on top of the TimingMonitor device server, which is continuously monitoring the system and warns about any problem it detects. A snapshot of the GUI can be seen in Fig. 4.

Figure 4: Timing monitor GUI.

The aspects that are monitored:

- EVR violation. All the EVRs can detect a problem in the fibre optics link it receives.
- Direct heartbeat. A heartbeat event is periodically sent by the EVG. A time out alarm is set by any EVR if it doesn't receive that event for a predefined time.
- Reverse heartbeat. Every EVR is instructed to send a heartbeat periodically up to the main EVR.
- EVR Alive. It checks the tango state of every EVR device server.
- Fanouts. It checks the status of the fanouts.

CONCLUSION

The development of the control of ALBA timing system has been quite a big effort. It has proved to be a very useful tool for commissioning, during which the fine tune of the timing of any of the key elements of the machine was possible with simply a two click operation. The save/open feature allowed the saving/restoring of any configuration of the whole timing system with a 3 click process. The permanent storage of all the settings in tango database allows a transparent recover after a system shutdown (i.e. for maintenance): all the settings are automatically restored on power up.

The timestamped logging feature also proved to be a valuable tool for commissioning, allowing to easily and quickly find the root of the problem when beam was suddenly lost.

The Fast Interlock system [3] has also proved to be the fastest system for interlocking the whole machine.

The control system has provided the users a very simple tool which abstract them from the big complexity of the underlying system.

As part of the tango project [6], all the code and its dependencies are free software and can be freely and easily reused by the community with relatively little effort.

CONTRIBUTIONS

Many people has contributed to this project. Specially remarkable is the outstanding effort in developing a big part of the software by R. Suñé before leaving Alba. Thanks to J. Pietarinen [2] for his excellent support for the devices he provided. Many thanks to O. Matilla and D. Beltrán, who were the main designers of the system itself. Thanks also to J. Jamroz for hardware support.

REFERENCES

[1] CELLS-ALBA: http://www.cells.es
[2] Micro-Research Finland: http://www.mrf.fi
[3] O. Matilla et Al, "The Alba Timing System. A know architecture with a Fast Interlock System Upgrade".
[4] Adlink: http://www.adlinktech.com
[5] Reichle & De-Massari: http://www.rdm.com
[6] Tango: http://www.tango-controls.org
[7] Swig: http://www.swig.org

STATUS OF ALMA SOFTWARE

T.C. Shen, J. Ibsen, R. Olguin, R. Soto, ALMA, Santiago, Chile

Abstract

The Atacama Large Millimeter /submillimeter Array (ALMA) will be a unique research instrument composed of at least 66 reconfigurable high-precision antennas, located at the Chajnantor plain in the Chilean Andes at an elevation of 5000 m. Each antenna contains instruments capable of receiving radio signals from 31.3 GHz up to 950 GHz. These signals are correlated inside a Correlator and the spectral data are finally saved into the Archive system together with the observation metadata. This paper describes the progress in the deployment of the ALMA software, with emphasis on the control software, which is built on top of the ALMA Common Software (ACS), a CORBA based middleware framework. In order to support and maintain the installed software, it is essential to have a mechanism to align and distribute the same version of software packages across all systems. This is achieved rigorously with weekly based regression tests and strict configuration control. A build farm to provide continuous integration and testing in simulation has been established as well. Given the large amount of antennas, it is imperative to have also a monitoring system to allow trend analysis of each component in order to trigger preventive maintenance activities. A challenge for which we are preparing this year consists in testing the whole ALMA software performing complete end-to-end operation, from proposal submission to data distribution to the ALMA Regional Centers. The experience gained during deployment, testing and operation support will be presented.

OVERVIEW

ALMA software [1] is a very complex system, developed by distributed teams across three continents. Every six months a new release takes place and once it is deployed at the Operation Support Facility (OSF), it is responsibility of the ALMA Department of Computing (ADC) in Chile [2] to take care of its maintenance. ALMA has scheduled for Sep 30th of 2011 to start operating in the so called "Early Science" mode. In preparation for such a big challenge, ADC has designed and implemented an entire support system around the core ALMA software in order to successfully achieve this important milestone.

In the following sections a description of the software verification and maintenance processes, together with the required infrastructure will be described.

ALMA OPERATION SUPPORT PROCESSES

Array Element Configuration

ALMA is composed of 66 antennas, photonics references, Correlators and control computers which need to be configured according to the science verification purpose. Due to ALMA still being under the construction phase, reconfiguration of the system is an activity run on daily basis.

ALMA observation software saves the system configuration variables in the Telescope Monitor and Configuration Database (TMCDB), a relational database built on top of Oracle R11. Each array element has an average of 1500 variables, in addition to the deployment information. It is unviable to do it manually, (as the first antenna has been done) and a template based approach has been developed, allowing to configure a new element in matters of minutes.

As soon as the number of antennas increases, it is hard to keep track of their status and deployment configuration. Several web based applications were developed to depict live deployment information.

Regression Tests

A new version of ALMA software is released every six months. The new release is deployed and verified on site together by North American, European and local software staffs. After the acceptance, the local software group is responsible for supporting and maintaining this new production release, together with guarantying the system's stability. Modifications in this operational release are driven either by software patches for bugs fixing or by adding new functionalities. A new weekly iteration of this release is deployed at the control computers in order to consolidate these modifications and a battery of regression tests is executed to guarantee system reliability.

The regression tests suite aims to verify the software functionality through the execution of the same tasks performed during science operations. Additionally, tests focused on the validation of the ALMA software deployment and are also executed.

The pass criteria applied for every test is simple and atomic. The idea is to avoid misinterpretation about the test completeness. For example:

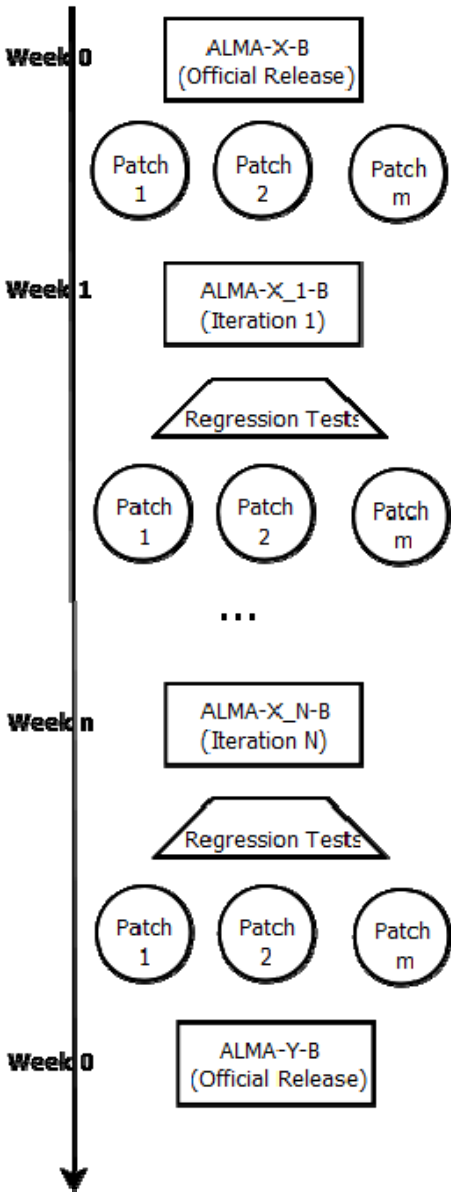


Figure 1: Regression Tests Workflow.

Description: *Create a manual array*
Test Execution: *Within the OMC, open the "Create Array" plugin. Create a manual array by selecting all available antennas, a Photonic Reference and a Correlator. (if another array was created, dispose of it previously)*
Pass Criteria: *A manual array has been created and all available antennas are assigned to it.*

Despite most of the tests being executed and the results verified manually, ADC has been working on the automation of the tests suite. A framework based on pyUnit has been implemented in order to allow the automatic execution of the tests and the verification of the passing criteria. The "Auto Regression Tests" framework takes the information from an XML configuration file, in which a test is defined. Every tests suit consists in: (1) cleanly initializing the system, (2) allocating the required resources, such as: the selection of available antennas, photonic references, and Correlators, (3) executing the specified tests, (4) releasing allocated resources and shutdown of the whole system, and (5) reporting the results. In cases, when the test suite collects scientific data, the data must be analyzed in order to verify the coherence of the observations. To cover this area, validation routines have been developed in collaboration with the science team, which in turn basically reduce the raw data by means of tools such as TPOINT, CASA, etc. Upon the event when a Boolean result is not easy to achieve visual inspections will be required.

In conclusion, regression tests provided an objective and reliable way to verify the system functionalities after the deployment of new software iteration. However, this process requires working continuously on the upgrade of test suites as scientific scripts evolve and new features being introduced in the system.

In the near future, the goal of this is to execute the same tests suite across all ALMA software development centres in order to verify and validate a fresh release.

End-To-End Tests

Aside from the weekly regression tests, which focus on the observation software, a complementary verification is done in parallel, which focuses on the end to end aspect. These tests start with (1) proposal submission phase, (2) proposal selection phase, (3) proposal scheduling and observation phase, (4) observation data replication from OSF to Santiago data centre and (5) data replication to ALMA regional centres.

Software Patch Workflow

The workflow starts with (1) bug detection and reporting, (2) debug and troubleshooting process, (3) patch preparation. (4) Control Board approval, (5) testing, (6) deployment in the operational environment, (7) and consolidation for the next software iteration.

ALMA uses JIRA as the main tickets reporting system. Together with JIRA, several custom data mining tools were developed to produce periodically statistics about downtime and the ticket associated. Other key process indicators are also generated in order to provide sufficient information to allocate efficiently the available resources.

Scalability Tests

These tests aim to validate in advance the capacity of ALMA software to deal with all the array elements when it is completed. Tests are focused on (1) diskless real-time machines booting process, (2) CORBA notification services, (3) TMCDB access during software start up, (4) monitoring system, and (6) dynamic resource allocation during observation.

Currently ALMA has 20 antennas integrated into the array. Scalability showed that the current software scales correctly up to 30 antennas, but problems start to occur

when the 40 antennas boundary is crossed, which is the number of antennas expected at the end of 2012.

ALMA SOFTWARE SUPPORT INFRASTRUCTURE

In order to successfully and efficiently perform the aforementioned processes, several computing infrastructures are required. In the following sections key infrastructures are depicted.

Virtualization

The virtualization technology has been the foundation to provide computing power in ALMA. Starting from 2009, several virtualization technologies have been tested, such as XEN, KVM. But the final choice was VMware ESX. The final decision was based on features of administration tools rather than a performance reason [3]. Virtualization is being used in several areas, not only in the testing and development environments but also in production environment, i.e: build farms, non real-time production servers (web, database, applications). Virtualization provides the flexibility and efficiency to deploy servers to accomplish dynamic ALMA operation needs.

After two years of production experience with virtualization, a new step has been taken: consolidate the hardware infrastructure in blade system. Two chassis with Dell M610 blade servers were procured and put into operation.

ALMA Software Build Farm

ALMA software is a four Gb. of source code which takes around ten hours to compile. Currently they are four different active versions being used. In order to support these four branches of software the concept of build farm had been introduced two years ago. Now there are six nodes configured to generate daily builds. Failures found during compilation are notified by email and remedy action is taken immediately.

Hudson has been chosen to be the main continuous integration tool. In addition of it, several custom scripts were created to generate change logs automatically from the code version control repository. These logs are useful for telling the nature of the modifications in the production branches and unsolicited modifications are investigated and reverted if there is no valid justification from the developers. Developers also benefit from the daily build in order to simplify the bug fixing and verification activities. Now having daily builds reduce the time to prepare a working environment to test patches.

Standard Test Environment

ALMA software was designed to run on top of a set of servers which are grouped as the minimum unit under the concept of Standard Test Environment (STE). [3]. In order to fulfil the requirement to commission and verify antennas in parallel, multiple STE were deployed at the OSF. Each STE has the same network subnets and addresses definition in order to guaranty the same environment to run ALMA software. To cope with the repetitive network addresses, virtual routing and forwarding (VRF) technology has been deployed and overlapping IP addresses can be used without conflicting with each other.

STE is also the platform being used to develop and test ALMA software under simulation. There is a total of 13 STEs across ALMA. Some of them are as simple as a set of three servers, and in the other hand, the one dedicated to control the array will eventually have up to 135 servers in it.

To administrate the configuration, installation and maintenance of such an amount of computers, an approach based on Puppet has been tested and validated. It is expected to be the official tool at the OSF soon. This tool will allow for an agile and efficient way to manage dynamic infrastructure.

Monitoring System

To operate the ALMA Observatory, aside from the previously mentioned array elements, it is likely important to make reference of the operation support infrastructure, such as, computer rooms, network devices, operation building facilities, power generators, etc. To guarantee a smooth and successful observation, both areas must work co-ordinately. By having two sites, the ALMA Operation Site (AOS) located at 5000 m., and the OSF located at 3000m over sea level, make the maintenance activities even more complex. In this context, the M&C (Monitor & Control) system plays a fundamental role on defining the live heath status of components of the entire Observatory. The design of the M&C must ensure a proper information distribution to users and specialists for decision making and must also provide alarms to trigger corrective actions, if required.

When ALMA reaches its full capacity with 66 elements in the array, there will be around 1 million monitor variables, which will generate information at a data rate of 40 GB/day. It's really a big challenge to maintain this M&C system, not only in the operational aspects but also solving the storage problem of such amount of data throughout time.

The ALMA Observatory M&C system is a three tiers software: (a) data acquisition layer, (b) data distribution and processing layer and (c) data display layer. Due to historical reason, the array M&C system was designed without considering the support infrastructure's M&C system and vice-versa. The first one is based on ACS BACI properties [1] and the second one is provided by an industrial solution provider. Both systems have components in the aforementioned three layers and works perfectly in the context where they were conceived. But unfortunately neither of both is flexible enough to interoperate with other. Therefore in order to integrate both systems, a third framework is introduced: Zenoss, this framework has several features which make it very suitable for this purpose: template oriented design for

quick devices configuration, vast range of monitoring protocols, powerful events engine, build-in notification services, Zope based core written in python and round robin database as main storage, etc.

By using Zenoss, as the interfaces which access both systems, it is possible to retrieve and consolidate information. A prototype implementation has demonstrated that is feasible to provide consolidated graphical users interfaces or dashboards according to user profiles, such as: array operators, facilities managers, instrumentations and software teams. Additionally, Zenoss provides a very convenient way to storage time series data by using round robin database (RRD). This automatically drops old data and eliminates additional maintenance work. Historical data saved in RRD allows providing trend analysis, characterize overall system behaviour and predict potential failures.

Currently array M&C and the implementation based on Zenoss are still under development. For infrastructure, the off the self system is currently installed and is being fine tuned.

CONCLUSION

ALMA has started to operate in "Early Science" mode since September 30th of 2011. And the preparation achieved in terms of software verification, has proven to be a key factor to achieve this huge milestone for the ALMA communities.

Testing infrastructure and simulation are important resources which are usually relegated to the least priority of astronomical projects. In our experience, these must be considered as part of the original design.

Regression tests, performed systematically, are essential to guaranty stable software and detect bugs introduced involuntarily by developers.

A prototype and semi-production monitoring system has been implemented and deployed. This tool produced relevant information for the maintenance department in order to schedule preventive and corrective activities.

Projects with a big number of elements must validate the scalability feature of its software design and implementation. We suggest that the earlier the better.

ALMA is the first astronomical project constructed and operated under industrial standards, therefore, it is important to include good practices from the industrial area.

ACKNOLEGMENT

The authors wish to acknowledge the contribution and cooperation of ALMA CIPT and especially to ALMA ADC Software Group, which works are reflected in this paper.

REFERENCES

[1] J. Schwarz, A. Farris, and H. Sommer, "The alma software architecture", Proceedings of SPIE, 5496, p. 190 (2004).

[2] V. Gonzalez, M. Mora, Other, "Fist year of ALMA site software deployment: Where everything comes together", Proceedings of the SPIE, 7737, p. 7731Z-77371Z-8 (2010).

[3] Zambrano, M., Arredondo, D., Other "Experience virtualizing the ALMA Common Software", ADASS XIX, 434, 477. (2010)

THE IMPLEMENTATION OF THE SPIRAL2 INJECTOR CONTROL SYSTEM

F. Gougnaud*, J.F. Denis, J.F. Gournay, Y. Lussignol, P. Mattei, R. Touzery, CEA-IRFU / Saclay, France
P. Gillette, C. Haquin, Ganil / Caen, France
J. Hosselet, C. Maazouzi, CNRS-IPHC / Strasbourg, France

Abstract

The EPICS framework was chosen for the Spiral2 project control system [1] in 2007. Four institutes are involved in the command control: Ganil (Caen), IPHC (Strasbourg) and IRFU (Saclay) and LPSC (Grenoble), the IRFU institute being in charge of the Injector controls. This injector includes two ECR sources (one for deuterons and one for A/q=3 ions) with their associated low-energy beam transport lines (LEBTs). The deuteron source is developed at Saclay and the A/q=3 ion source at Grenoble. Both lines will merge before injecting beam in a RFQ cavity for pre acceleration. This paper presents the control system for both injector beam lines with their diagnostics (Faraday cups, ACCT/DCCT, profilers, emittance-meters) and slits. This control relies on COTS VME boards and an EPICS software platform. Modbus/TCP protocol is also used with industrial devices like power supplies and Siemens PLCs. The Injector graphical user interface is based on EDM while the port to CSS BOY is under evaluation; also high level applications are developed in Java. This paper also emphasizes the EPICS development for new industrial VME boards ADAS ICV108/178 with a sampling rate ranging from 100 K Samples/s to 1.2 M Samples/s. This new software is used for the beam intensity measurement by diagnostics and the acquisition of sources.

THE SPIRAL2 INJECTOR

The Spiral2 Injector Control System is globally managed by CEA-IRFU.

The Spiral2 Injector is composed of 2 beam lines, one (LEBT1) developed at LPSC Laboratory, Grenoble and the other (LEBT2) at the IRFU Institute, CEA Saclay. Both lines will merge at the end of 2012, before injecting the beam into an RFQ cavity.

The beam line in Grenoble was commissioned in 2009. It is controlled by Epics but the Ion source has been controlled by a Labview PC for a long time. The goal is to control all the equipment from Epics. So, a replacement by Epics is in progress. All Labview interfaced parameters are implemented in a soft Epics database that uses the Labview-CA gateway.

On the LEBT2 at Saclay, the control system is being finalized. On 15th September, 2011, a 7mA proton beam was conducted to the end of the line, with 85% transmission, i.e. 5.5 mA at the end of the line.

The Saclay proton/deuteron source control system has been done in Epics. It controls several pieces of equipment: a magnetron, a pulse generator, an impedance adaptor, an extracting beam High Voltage power supply, a gas injector controller and a repelling electrode power supply.

*francoise.gougnaud@cea.fr

SPIRAL2 EPICS PLATFORM

Software Platform

The goal of this platform is to install all the PCs and VMEs of the Spiral2 accelerator in the same manner. This platform provides a software development model that each Epics developer has to use to build the different Spiral2 applications. In this way, we have got a homogeneous control system on the whole Injector. At present this platform uses Epics 3.14.12, VxWorks 6.8 and Linux RHEL5 on PCs. Modbus/TCP protocol is used to interface PLCs, power supplies and some beam diagnostics such as beam profilers.

A Subversion (SVN) repository is available at Ganil to manage the software development [2].

Hardware Platform

The Spiral2 Injector standard VME is composed of an Emerson Motorola MVME5500 and NEXEYA ADAS VME boards [3]:

- ICV150 with 32 ADCs, 16 bit resolution and 30K Samples/s
- ICV714 with 16 DACs, 12bit resolution
- ICV196 including 96 binary input/output channels
- ICV108, a controller board with a 4 M bytes RAM
- ICV178 with 8 ΣΔ ADCs, 16 bit resolution, 1.2 M Samples/s.

IRFU has developed the Epics drivers for these boards. Three VMEs control the LEBTs Epics control system and one specific VME is dedicated to controlling emittance-meters.

POWER SUPPLIES

This software was designed by Ganil [4]. All the power supplies are of Hazemayer type and are current power supplies. They are connected to the Ethernet field bus and are accessed via the Modbus/TCP protocol. The Epics driver, provided by Mark Rivers from the University of Chicago, is used to control these power supplies. At present on the line LEBT1, in Grenoble, there are 15 power supplies (1 Sextupole, 1 dipole, 1 solenoid, 2 triplets of quadrupoles, 6 steerers). On the LEBT2 line, at

Saclay, there are 20 power supplies (2 dipoles, 2 solenoids, 7 quadrupoles, 9 steerers).

"FAST ACQUISITION" AND FARADAY CUPS, ACCTS AND DCCTS

The beam intensity of the Spiral2 Injector is measured in 2 ways. One way is beam destructive measurements with five Faraday cups [5] in the Low Energy Beamline Transfer, and beam non-destructive measurements by one ACCT and two DCCTs in the Low and Middle Energy Beamline Transfers. The beam runs in pulsed or CW mode.

At the beginning of the project, we developed a beam current measurement based on the VME ICV150 board (30 K Samples/s) but this acquisition wasn't precise enough with pulsed beam and couldn't display the shape of the pulse.

The principle for this acquisition is to synchronize the acquisition at the pulse start [6], to buffer the digitalised data, to display the pulse shape and to calculate an average value for each pulse.

The solution relies on the VME ADAS ICV108 and ICV178. These boards are described more precisely in [7]. Both boards communicate via the VME I/O P2 lines. The ICV108 includes an external trigger and a RAM buffer of 4 M bytes. This acquisition uses the RAM in "Flip/Flop" mode.

At present the Epics driver permits the use of two boards ICV178, i.e. 16 channels. To optimize running with regard to the 4 M bytes RAM and the sampling frequency, only the channels, corresponding to the equipment used for the intensity measurement, are sampled. The operator chooses the configuration on the display and the driver changes the configuration "on-fly".

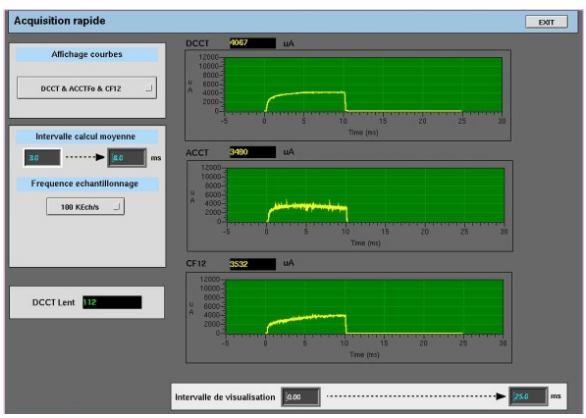


Figure 1: EDM display with one ACCT, one DCCT and one Faraday Cup.

This driver provides compressed (averaged) data, if necessary, to the applications, to speed up the visualization of graphs. The configuration "compressed" or "not compressed" is specified in the Epics database.

On the display (fig. 1), the operator has to give the interval of visualization and the interval dedicated to the average.

This fast acquisition is also used by Hall probes and emittance-meters.

BEAM PROFILERS

Profilers permit to measure and visualize the beam dynamics. In the LEBTs, these profilers are secondary emission profilers. This equipment and its control are completely managed by Ganil. The communication with the electronics is based on Modbus/TCP. An Epics IOC application is used to configure measurements and to calculate average and offset subtraction. A JAVA server task [8] provides supplementary calculations to those carried out by the IOC application to the client applications. These new calculations are supplied as Epics Process Variables that are surface, signals sum, gravity center, standard deviations and widths. They are done for an area of a horizontal or vertical profile according to the requested configuration.

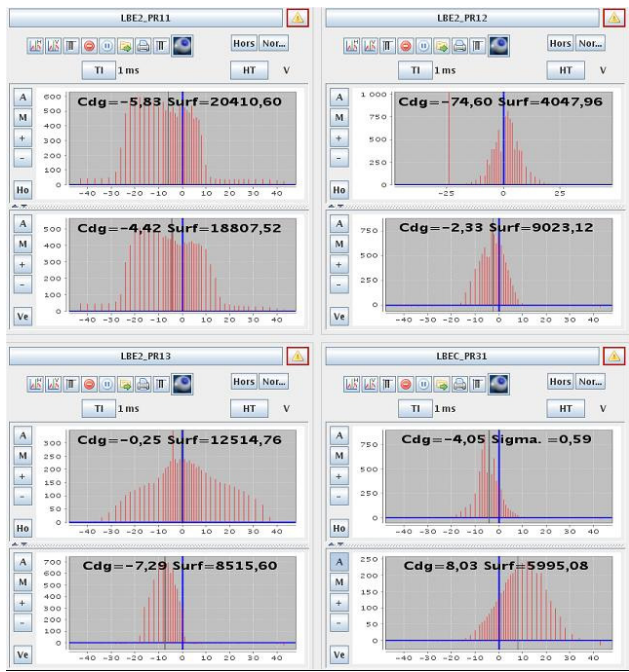


Figure 2: Beam profiles display, JAVA application using XAL framework.

EMITTANCE-METERS

The transversal emittance is the cartography of the transverse angles of the particules of the beam. It provides information on the divergent or convergent aspect of the beam and is used to measure the focal of the beam to study the propagation of the particles and to tune the beam.

At present a pair of emittance-meters is installed on each of the 2 beam lines. These emittance-meters and their control are carried out by IPHC Institute in Strasbourg.

A specific VME controls the 3 pairs of emittance-meters of the Injector. It is composed of the standard

VME boards: one ICV150 to acquire temperatures, a pair of ICV108/178 dedicated to faster acquisition, one ICV196 binary input /output board used for selection and configurations for hardware, one ICV714 to generate voltage ramps and besides an ISEG board to generate high voltage and a brushless motor board OMS MaxV to move the actuator of the measurement head.

When the emittance measurement is requested, the program gets the user parameters and waits for validation of the PLC process. Then, the program applies the maximum voltages requested to the plates of the measurement head and comes in a loop made up of 3 sequences: positioning, voltage ramp setting and current reading. For each position during the move, the voltages on the plates are decreased until an electrical field equal to zero is reached. Then, the field is inversed and the sequence position, voltage ramp, current reading is repeated. For every step of the voltage ramp, the current created in the Faraday cup is recorded.

For the human operator interface, there are 3 levels of EDM displays: data catching, configuration and debugging. A 3D visualisation is necessary. Therefore, a Java HMI was developed and recently a CSS BOY display has been created (fig. 3).

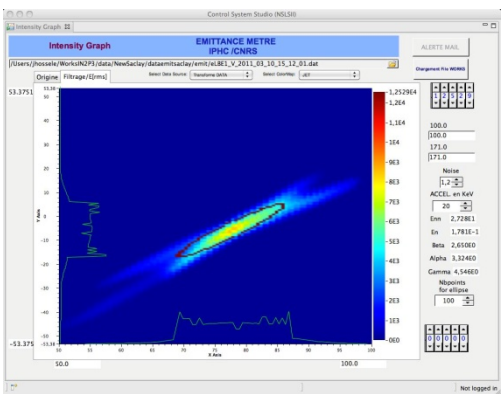


Figure 3: Emittance-meter HMI in BOY.

SLITS

The objective of the slits is to remove a part of the halo surrounding the heart of the beam. A pair of thermocouples is integrated into each slit to monitor the temperature that is controlled by an Epics IOC through the ICV150 ADC board. The control of the motor driver is based on Modbus /TCP in the same way as the power supplies.

PLCS FOR INTERLOCK AND VACUUM

Siemens S7 PLCs were chosen by the Spiral2 project. Interlock PLCs provide safety of the pieces of equipment on the beam line at Saclay and at Grenoble. When these 2 lines join at Ganil, only one PLC will ensure the two beam lines interlock. Vacuum is controlled by one PLC at Saclay and one PLC at Grenoble. The PLCs are connected to Ethernet and supervised by a Muscade station and with Epics EDM HMIs. Muscade is a proprietary CEA SCADA whose objectives are supervision, diagnostics and distant maintenance. It runs on Windows PCs.

Protocol Modbus/TCP is used for the communication between Epics IOCs and PLCs.

The PLC can run in “local” or “distant” mode. This mode is read by Epics. When the PLC is in local mode, it memorizes the commands for Epics.

HOST-BASED PERL APPLICATIONS

A set of applications running on the Linux OPIs have been developed based on Linux. They are written in Perl and they use the cap5 Channel Access library provided by Andrew Johnson from ANL. A wrapper above cap5 has been designed to hide the details of the implementation. The interest of this approach has several aspects compared to a similar development made in SNL on the VME target for example: intrinsic power of the Perl language, ease of developments of applications based on an interpreted language, availability of a simple debugger. The performance is not a real issue for the type of applications aimed (fast acquisition buffers of several hundred of bytes can be read at 10Hz, processed and stored on disk without any problem).

The following applications have been developed: correlation of parameters for beam tuning, ECR sources beam spectrum acquisition, a thumbwheel generic tool for parameters fine setting and several utilities functions.

ACKNOWLEDGEMENTS

Special thanks have to be addressed to Jean-Michel Joubert for the cabling and Arnaud Roger, Tom Joannem and Cedric Peron (IRFU/SIS) for their development on PLCs.

REFERENCES

[1] E. Lécorché et al. “Development of the future Spiral2 control system“, Icalepcs 2009, Kobe, Japan.
[2] D. Touchard et al. “The Spiral2 command control software organisation and management”, Icalepcs 2009, Kobe, Japan.
[3] http://www.adas.fr, NEXEYA ADAS company
[4] D. Touchard et C. Haquin. “A Modbus/TCP-based Power Supply Interface”, PCaPAC08, Ljubljana, Slovenia.
[5] C. Jamet et al., “Injector diagnostics overview of Spiral2 accelerator”, Dipac 2007, Venice, Italy
[6] P. Ausset et al, “SPIRAL2 Injector Diagnostics”, Dipac 2009, Basel, Switzerland.
[7] F. Gougnaud, P. Mattei “The first steps of the beam intensity measurement of the Spiral2 injector”, Icalepcs2009, Kobe, Japan.
[8] Spiral2 control command:first high level Java applications based on the Open-Xal library P. Gillette et al, Icalepcs 2011.

A READOUT AND CONTROL SYSTEM FOR A CTA PROTOTYPE TELESCOPE

I. Oya, U. Schwanke,
Institut für Physik, Humboldt-Universität zu Berlin, Berlin, Germany
B. Behera, D Melkumyan, T. Schmidt, P. Wegner, S. Wiesand, M. Winde,
Deutsches Elektronen-Synchotron, DESY, Zeuthen, Germany
for the CTA Consortium*

Abstract

CTA (Cherenkov Telescope Array) is an initiative to build the next generation ground-based γ-ray instrument. The CTA array will allow studies in the very-high-energy domain in the range from a few tens of GeV to more than a hundred TeV, extending the existing energy coverage and increasing by a factor 10 the sensitivity compared to current installations, while enhancing other aspects like angular and energy resolution. These goals require the use of at least three different sizes of telescopes. CTA will comprise two arrays (one in the Northern hemisphere and one in the Southern hemisphere) for full sky coverage and will be operated as an open observatory.

A prototype for the Medium Size Telescope (MST) type is under development and will be deployed in Berlin by the end of 2011. The MST prototype will consist of the mechanical structure, drive system and active mirror control. Four CCD cameras and a weather station will allow the measurement of the performance of the instrument. The ALMA Common Software (ACS) distributed control framework has been chosen for the implementation of the control system of the prototype. In the present approach, the interface to some of the hardware devices is achieved by using the OPC Unified Architecture (OPC UA). A code-generation framework (ACSCG) has been designed for ACS modeling. In this contribution the progress in the design and implementation of the control system for the CTA MST prototype is described.

THE CTA ARRAY

Very High Energy (VHE, energies > 100 GeV) γ-rays are produced in both galactic and extra-galactic objects like pulsars, pulsar wind nebulae, supernova remnants and active galactic nuclei. In recent years, thanks to the success of Imaging Atmospheric Cherenkov Telescope (IACT) instruments like H.E.S.S., MAGIC and VERITAS the field has experimented a major boost, with more than 100 sources already established as VHE emitters. The technique used by IACTs is based on the fact that VHE γ-rays interacting with the atmosphere produce electromagnetic showers, whose charged components generate Cherenkov light in a narrow cone. An IACT consists of a reflector that concentrates the Cherenkov light in a camera composed of sensitive photo-detectors, usually photomultiplier tubes (PMTs). The measured light distribution in the camera of each recorded event allows to reconstruct the arriving direction, energy and type of primary particle which caused the particle shower. In systems of telescopes the individual images can be combined to provide stereoscopic information, enhancing the resolution and background suppression.

The goals of CTA require the use of more than 50 telescopes of three different sizes. A few large size telescopes (LST), with dishes of 23 m diameter, are designed to achieve a good sensitivity in the energy domain from a few tens of GeV up to about a hundred GeV. In order to achieve a good sensitivity at energies above a few TeV a large number of small-size telescopes (SSTs) will be deployed. For the core energies a considerable number of MSTs, of 10-12 m diameter each, will be used.

CTA will comprise two arrays (one in the Northern hemisphere and one in the Southern hemisphere) for full sky coverage and will be operated as an open observatory. By extrapolating the distribution of known VHE objects, it is expected that CTA will detect around 1000 new objects. Details on the science motivation, design concepts and expected performance of CTA are provided in [1].

THE MEDIUM SIZE TELESCOPE PROTOTYPE

A prototype of one of the design concepts of the MST is under development [2] and will be deployed in Berlin (Adlershof Campus) by the end of 2011 or the beginning of 2012. The main goal of this prototype is to test a design of the mechanical structure and drive system, but other prototype instruments like the Active Mirror Control (AMC) units will also be tested. No PMT camera will be installed on this prototype.

The MST prototype has a Davies-Cotton type reflector with diameter of 12 m, and a focal length of 16 m. The design of the telescope structure can be seen in Fig 1. Besides the mechanical structure, the prototype will consist of a dummy camera of 2.5 tons (to resemble the effect of the real PMT camera that will be in the final MST units), the drive system, active mirror control, four Charged-Coupled Device (CCD) cameras and a Weather Station (WS), plus some additional sensors designed to test the behavior of the structure. Time stamps of events will be obtained either via the Network Time Protocol (NTP) or a Global Position System (GPS) receiver. Emulated data sources will be

* see http://www.cta-observatory.org/ for full author and affiliation list

employed to resemble a realistic operation scenario of the prototype. Details of individual devices are provided later in this document. A full size prototype of one-quarter section of the dish has been constructed. The purpose of this prototype is to test the fabrication methods, costs, and to provide a *test bench* for mounting the mirrors.

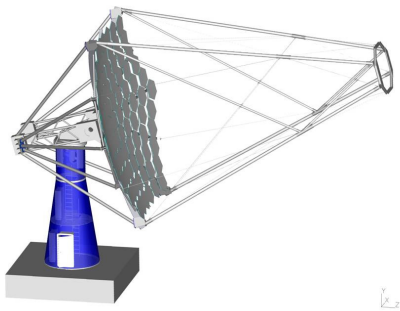


Figure 1: View of the design of the mechanical structure of the MST, composed by a quadrupod (camera support), a dish structure, a counterweight, the "head" (connecting the dish and the tower), and the tower with the foundation.

CONTROL SOFTWARE FOR THE MST PROTOTYPE

The MST prototype is the first opportunity inside the CTA consortium to integrate a considerable number of hardware devices and to exercise the control software. Each individual CTA telescope will be a complex system that has to operate in synchronization with the other telescopes and auxiliary devices. The CTA consortium will allow flexible observation scheduling by including the possibility of fragmenting the arrays in several sub-arrays.

Typically, a control software for an installation of the size of CTA should allow a flexible way of accessing the hardware. Engineers would require expert modes of accessing the devices for testing purposes, while astronomers will require a framework to operate an array composed of several dozens of telescopes, typically without being familiar with the complex details of the devices. The mentioned constrains fit naturally in the distributed computing paradigm, and in particular in the Object Management Group (OMG) CORBA specification, the most successful distributed object platform.

The ALMA Control Software [3] is a framework on top of the operating system that provides a complete environment and structures at the base of application software developments. ACS is meant to be a general system, based on CORBA and the C++, Java and Python languages. The core of ACS is based on a distributed component model, with ACS components implemented as CORBA objects in any of the supported programming languages. ACS Components are the base for high level control entities and for the implementation of devices such as a telescope drive system. ACS provides common CORBA-based services such as logging, error and alarm management, configuration database and lifecycle management, while hiding the complexities of CORBA programming. ACS is developed by the ALMA project and the main responsibility for the subsystem is at European Southern Observatory (ESO).

Due to the similar level of complexity and requirements of ALMA and CTA, ACS is currently being considered by the CTA consortium members to serve as the control middleware. It has been decided to use ACS for controlling the MST prototype because it will provide an excellent platform to test the behavior of ACS with CTA hardware, while serving as a learning platform.

CTA will be composed of a large number of devices that in principle might use very different hardware interfaces (serial lines, Ethernet, CAN-bus, etc.) Large experiments benefit by setting a standard way of accessing the hardware, either by defining a standard hardware interface like in ALMA (CAN-Bus) or by setting a lower level software layer like, for example, OPen Connectivity-Unified Architecture (OPC UA). For CTA it is in general preferred to select devices using Ethernet interfaces, and to set OPC UA servers for those devices requiring a different interface. Some preliminary studies and prototype developments have been taken place at LAPP[1] [4] in order to illustrate the implementation of the OPC UA standard. In particular these implementations concerned some electronic components and related functionalities devoted to the online security control of the H.E.S.S. 2 [5] camera. These studies have confirmed the homogeneous way of connecting electronic boards via the low software layer of the OPC UA and the standard way of integration in the ACS environment (by means of the DevIO mechanism in the ACS terminology). A similar approach has been adopted for the MST prototype: A standard way of deploying OPC UA servers has been investigated, and an ACS access method (a Java DevIO class) is under development with the support of the ACS development team from ESO, complementing the existing C++ DevIO for OPC UA [6].

The ACS components for the control of the MST prototype are being designed using an Unified Modeling Language (UML) based code generator. The UML models are created using MagicDraw UML Standard v.17.0 on Linux, which is used to create the input required by the ACS code generation framework (ACSCG) [7]. From the UML model, ACSCG generates the interface definition language (IDL), configuration database files and Java implementation skeleton. The Java skeleton can be extended to provide an OPC UA client functionality, in order to communicate with an OPC UA server. In order to model the OPC UA servers the OPC UA Java Software Development Kit (SDK) v. 1.1.0-975 is being used.

A software development, integration and test environment has been deployed in DESY, composed by two Dell PowerEdge R510 machines with 12 fast disks, 12 Nehalem

[1]Laboratoire d'Annecy-le-Vieux de Physique de Particules-http://lapp.in2p3.fr/

Cores, 36 GB RAM memory and 10 Gbit Ethernet interface using the same layer level 2 switch. A RPM based installation of 64 bit ACS releases has been developed. An archive for permanent storage of the prototype measurements is under development, which will also serve as a *test bench* for the CTA archive. The DESY data-center will be connected to the MST prototype site, ca. 15 km away, by a broadband connection.

PROTOTYPE INSTRUMENTATION

Drive System

The drive system of the MST prototype is designed to resemble the expected operation modes of the CTA telescopes, allowing pointing of the prototype to any position and to track any astronomical object. The telescope will operate with two motors, one for azimuth and one for elevation. As a first instance to test the drive concept, a test stand was build to evaluate the design (see Fig 2).

The drive system of the prototype is composed, at a lower level, of the control of 6 drives (2 for azimuth and 6 for elevation) communicating via a *Bosch-Rexrot* programmable logic controller (PLC). The PLC operates VxWorks and hosts an OPC server. Earlier versions of the PLC firmware provided an OPC Data Access (DA) specification server, with plans to be replaced with an OPC UA server. As OPC DA is only working for Microsoft Windows environments, and for CTA Linux based software will be used, a solution was developed to deal with the limitation imposed by OPC DA, consisting in the deployment of an OPC DA to OPC UA wrapper software as interface to Linux software (*MatrikonOPC UA Wrapper for COM OPC Servers*). At the time of writing this document, the *Bosch-Rexroth* PLC firmware was upgraded to provide an OPC UA server that was under evaluation. The next step will be to create an ACS component for the drive system which will use the OPC Java DevIO class mentioned before.

After the ACS component has been tested and deployed, the drive control software will be used to interface the drive test stand. When the prototype is deployed a first phase of commissioning will take place, when the control software will be used to tune the system by using fiducial marks. In a later phase, the control system will be used to track astronomical objects and to check the pointing with the CCD cameras.

The CCD Cameras

The most important tool to check different aspects of the MST prototype will be a set of four CCD cameras. One goal of these CCD cameras will be to test the structural design of the prototype, and, with the help of a WS, the effect of the temperature, wind and other environmental factors. Additionally, the CCDs will be used to perform mirror adjustments, and will allow for measurements of the optical point spread function. The output images will be stored in the standard astronomical FITS data format. Excluding the emulated data sources, the CCD cameras will be the source of the largest fraction of data volume of the prototype.

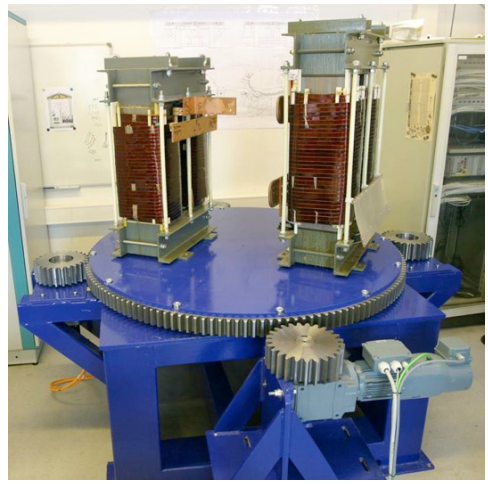


Figure 2: The MST prototype test stand.

The chosen CCD camera model is a Prosilica GC 1350 which has a resolution of 1.4 Mpix (1360 x 1024) and a pixel size of 4.65 μm. These CCD cameras are interfaced via GigE Vision interface (allowing up to 1000 Mbit/s on Gbit Ethernet). The Allied Vision Technologies (AVT) PvAPI SDK allows to control and capture images from GigE Vision CCD cameras in a Linux environment. It is accessible by most programming languages such as C++ and Java (via JNI). The evaluation of this software has been positive, working correctly in the test machines in both implementation languages. The next step involves the implementation of an ACS component to operate the CCDs.

The Active Mirror Control

The design of the CTA telescopes makes use of tessellated reflector composed of individual mirror facets (see [8] for details). Each individual mirror facet will be attached to an AMC unit which has the functionality of allowing a perfect mirror alignment. This will allow online re-alignments of the LSTs reflectors, where the deformation caused by the weigh of the telescope will cause misalignments depending on the telescope elevation. It will also allow, for larger timescales, to perform re-alignments in MSTs and SSTs. The dish of the MST prototype will be completely covered by a mixture of real and dummy mirrors. Several AMC units of two different AMC design concepts will be installed in the prototype (see [1] for details in the design concepts). One type of unit communicates via XBee radio modules, creating a Wireless Personal Area Network (WPAN) that is accessed via a XBee receiver connected to a PC via USB or RS-232 serial interface. The other type of unit is interfaced via CAN-Bus, accessed via a Ethernet-CAN-Bus gateway.

From the MST prototype control design point of view the idea is to unify the higher level interfacing to both unit types. This can be achieved with a higher lever ACS com-

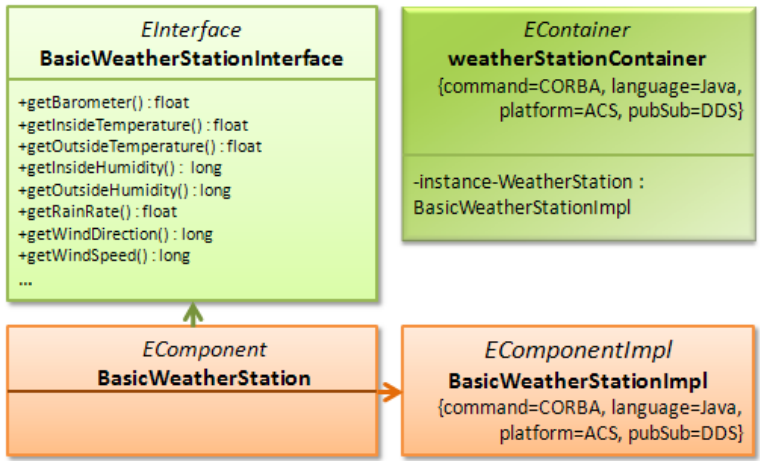


Figure 3: UML class diagram of an ACS component for the weather station for the prototype, modeled with MagicDraw.

ponent which sends the common instructions for the alignment procedures, acting as client for the lower level ACS components that will be different for each of the two AMC concepts.

For the AMC units using the XBee communication a preliminary version of a Python ACS component which uses the *Pyserial* library has been developed. Real movement of the AMC actuators was tested by using the ACS *Object Explorer* generic client. The movement of an AMC unit installed in the "quarter-dish" has also been recently tested.

The Weather Station

A WS has been acquired and will be installed nearby the MST prototype to continuously monitor the weather parameters, allowing correlation with the behavior of the structure of the MST with changes in the environmental status like, for example, the wind speed or temperature.

The chosen WS model is a *Davis Vantage VUE*. This WS is able to measure the wind speed and direction, as well as other quantities with the required accuracy and measurement rate. The instrument is composed of an outdoor unit communicating via a WPAN with an indoor unit, including a data-logger with some limited internal storage capacity and equipped with a RS-232 serial interface.

The WS is the least complex component of the MST prototype and for that reason has been the first device used to experiment with previously mentioned software development procedures. Starting from the lower level, an OPC UA server has been implemented with the OPC UA Java SDK which uses the RXTX library to communicate with the WS *datalogger* via the serial line. In a higher lever, a Java ACS component has been implemented using the UML-model based code generation framework (see Fig. 3), and tested with the ACS *object explorer*.

CONCLUSIONS

The MST prototype will act as the first realistic *test bench* for the control software for the CTA array. Equipped with complex systems like the drive system and AMC, the MST prototype will allow the evaluation of the ACS control middleware while the hardware design concepts are being tested, providing input and expertise to the software control developers of CTA. The use of tools like the UML code generation and the OPC UA servers, already tested the relatively easy system of the WS, will eventually allow the setting of a efficient framework for the CTA array control while optimize the productivity.

ACKNOWLEDGMENTS

We thank the excellent support of ACS experts from ESO Garching, in particular G. Chiozzi and H. Sommer. We want to acknowledge the help of A. Lopatin by teaching us ACS. We gratefully acknowledge support from the agencies and organisations listed in this page: http://www.ctaobservatory.org/?q=node/22

REFERENCES

[1] CTA Consortium, "Design Concepts for the Cherenkov Telescope Array", arXiv:1008.3703, 2010.

[2] E. Davis, et al. "Mechanical Design of a Medium-Size Telescope for CTA". CTA internal note.

[3] G. Chiozzi, et al., "The ALMA Common software: a developer friendly CORBA based framework", SPIE Procs. 5496-23 (2004).

[4] Le Flour, J.L. Panazol (LAPP/IN2P3). "OPCUA for CTA readout", CTA internal communication, Heidelberg Feb. 2011

[5] http://www.mpi-hd.mpg.de/hfm/HESS/

[6] Di Marcantonio, P., et al. "Evaluation of Software and Electronics Technologies for the Control of the E-ELT Instruments: a Case Study", in these proceedings.

[7] N. Troncoso, et al. "A Code Generation Framework for the ALMA Common Software", SPIE Procs. 7740-121 (2010).

[8] A. Föster, et al. "Mirror Development for CTA". Proc of the 32^{nd} International Cosmic Ray Conference, Beijing, China, 2011.

CONTROLS SYSTEM DEVELOPMENTS FOR THE ERL FACILITY*

James Jamilkowski#, Zeynep Altinbas, David Mark Gassner, Lawrence T. Hoff, Prerana Kankiya, Dmitry Kayran, Toby Allen Miller, Robert H. Olsen, Brian Sheehy, Wencan Xu,
BNL, Upton, Long Island, New York

Abstract

The BNL Energy Recovery LINAC (ERL) is a high beam current, superconducting RF electron accelerator that is being commissioned to serve as a research and development prototype for a RHIC facility upgrade for electron-ion collision (eRHIC). Key components of the machine include a laser, photocathode, and 5-cell superconducting RF cavity operating at a frequency of 703 MHz. Starting with a foundation based on existing ADO software running on Linux servers and on the VME/VxWorks platforms developed for RHIC, we are developing a controls system that incorporates a wide range of hardware I/O interfaces that are needed for machine R&D. Details of the system layout, specifications, and user interfaces are provided.

INTRODUCTION

Since the BNL ERL was conceived as an R&D testbed for a future electron-Relativistic Heavy Ion Collider (eRHIC) upgrade, many elements of the necessary Controls System have been designed and implemented. A 20 MeV superconducting 5-cell 703 MHz RF cavity that will accelerate and decelerate the electron beam passed its first milestone of the Cold Emission Test (CET) within the last two years [1]. Aside from further 5-cell RF measurements, Fundamental Power Coupler standing-wave RF conditioning was completed this summer for the superconducting gun FPCs at a power up to 125 kW (CW) and 250 kW (pulsed).

In 2012, the first ERL Gun-to-5-cell-cavity (G5) test where ~1 μA electron beam will be generated using the gun/laser assembly and accelerated in a simplified, linear version of the ERL layout is expected to begin. The ultimate goal for the completed facility will be to verify efficient energy recovery while running CW reaching electron beam average currents on the order of ~0.5 Amperes [2].

INFRASTRUCTURE

The ERL has benefited from a long legacy of hardware and software development at the Collider-Accelerator Department (C-AD). Where possible, we have leveraged existing interfaces that utilize our C++ Accelerator Device Object (ADO) based software running on both VME/VxWorks and LINUX Red Hat machines. Services such as archiving, logging, and storage are shared between ERL and the other parts of the C-AD complex, though they reside on a separate Ethernet subnet as shown in Fig. 1.

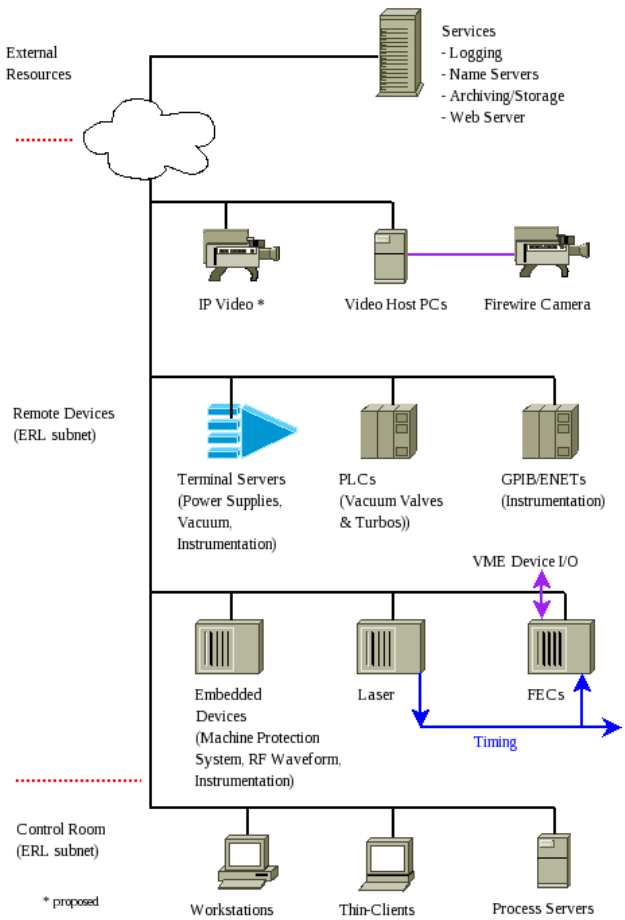


Figure 1: ERL Controls network-based architecture, including existing and proposed elements.

For legacy reasons, the Cryogenic System resides on an independent controls system and subnet of the Ethernet network. For the purposes of ERL, selected Cryogenic data is transferred to the Controls System at a rate of 0.2 Hz. Details of the Machine Protection System (MPS) can be found in ref. [3].

INSTRUMENTATION

Beam Position Monitors

ERL beam position monitor signals will be processed using Libera Brilliance Single Pass modules from Instrumentation Technologies. Five of these devices will be used during the first ERL beam tests scheduled for early 2012.

*Work supported by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the US Department of Energy.
#jpj@bnl.gov

The Libera device includes an ARM processor, on which an embedded configuration of LINUX runs, providing access to the functionality of the device via a vendor-supplied library and collection of kernel modules. The vendor also provides a cross-development environment, to allow software development on a conventional desktop LINUX system.

Using the cross-development environment and the library's programming interface, software was developed which runs on the Libera's processor, providing remote access to the device's functionality. The code base is identical to the code base used for VME or other embedded systems used in the facility. In this manner, each Libera device appears, from the control system perspective, to be just another network-attached embedded controller.

Each device will provide a TTL signal that will be used by the MPS to interlock when position readings stray from acceptable values.

Beam Loss Monitors

Four types of electron beam loss monitoring detectors are employed at the ERL: Photo-Multiplier Tubes (PMTs), Ion Chambers (IC), Heliax cables, and PIN Diodes.

The PMTs & ICs will provide current signals with respect to loss levels that will be processed by custom VME based amplifier/integrator/detector cards currently used in the RHIC beam loss system. These cards provide analog signals to standard VME ADC modules for monitoring and a TTL output for beam loss interlock level to the MPS. CAEN 24 channel HV VME controller modules will supply individually controllable references for the PMT high-voltage bias allowing each of the 14 PMTs to be "tuned" for a desired response. The 8 ICs are biased collectively by a manually adjusted supply.

The 16 Heliax cable detectors each have their own pair of bias supply and integrator module, identical to what is currently used for beam loss detection in the Alternating Gradient Synchrotron (AGS). Analog output loss data will be digitized using VME ADC modules, and control will be effected using a VME digital output module.

Eight Bergoz BLM PIN diode modules have been employed at the ERL using a SIS VME scaler module for data acquisition and a VME digital output module for calibration pulse control.

One TTL signal will be provided to the MPS that will indicate the status of the beam losses relative to a trip threshold.

Beam Profile Monitors and Cameras

A series of actuated profile monitors from Radiabeam Technologies will be used to gather information regarding the electron beam profile from light gathered from both YAG and Optical Transition Radiation (OTR) screens. Standard VME digital I/O modules will provide the insertion controls for each multi-position instrumentation package, and image capture will be performed using a setup that includes a LINUX host PC with a IEEE1394 Firewire interface to the attached camera. The image gathering and analysis software has been well-documented in ref. [4] using our custom ADO toolset.

To date, LINUX software Firewire support issues have prevented the active use of more than one camera per host PC through our standard software. As a potential alternative, we will evaluate a Gigabit Ethernet camera for use at the ERL. The feasibility of using low-cost, "lightweight", dedicated host PCs instead of the more expensive multi-role servers that have been previously used will be explored as well.

Multi-axis lens controller modules from Image Labs will be used to adjust the picture for each camera. LINUX ADO software provides remote access to the device through a serial terminal server for up to two cameras per unit.

A TTL insertion status signal from each profile monitor will provide an input with which to interlock the MPS under high average beam current conditions in order to prevent damage to the YAG flag screens.

Pepper Pot

Electron beam emittance measurements will be acquired and presented via a remote interface that is currently being designed. One possible implementation of the system would make use of existing PC-based data acquisition and would require further development of custom software that has been created to perform similar measurements for the Electron Beam Ion Source (EBIS) project. Actuator controls will make use of VME digital I/O modules.

Halo Scraper

Six scraper jaws will be used in the injection straight and zig-zag sections to remove the electron beam halo in support of measurements under limited average beam current conditions. Each jaw is actuated using stepper motor controllers interfaced with VME Front-End-Computers (FECs) that provide position control, position readings, and limit switch detection. The motion controller interface software was created for previous projects.

Signals derived from controller status relays are expected to provide the necessary MPS inputs in order to protect the scraper jaws from intercepting high average beam currents while running in a high intensity mode.

Faraday Cups

The halo scrapers will perform double-duty as Faraday Cup pick-ups. Three additional Faraday Cup beam dumps will be installed for the measurement of beam current during the initial G5 tests, progressing to six in the zig-zag portion of the G5 injection transport. VME analog modules will be used for the data acquisition to the Controls System through custom Faraday Cup amplifier cards installed in a Eurochassis. External triggering of the sampling will be performed using a VME delay module that was created for past projects.

Integrating Current Transformer

The Bergoz BCM-IHR ICT will monitor the performance of the accelerator and alert the MPS should the beam current exceed the maximum allowed for a given operational mode. Gain, calibration pulse parameters, and signal polarity control will be provided using VME digital output modules, and triggering of the integration will be executed using a VME trigger delay board.

Methods for transmitting the ICT statuses to the MPS are currently being reviewed.

Differential DC Current Transformer

A platform is in the planning stages that has the goal of providing beam current readings from matched Bergoz NPCT transformers in the injection and beam dump transport while comparing both signals with a reference generated by a current source. A loss of current between the two transformers over a desired threshold would trigger an input to the MPS as an indication of significant beam losses. The merits of two basic options are under review: PLC-based sampling with interlock outputs, or a Xilinx processor paired with ADC and digital output daughter cards. The latter platform has been used for an increasing number of RF [5] and Instrumentation projects at C-AD in recent years, each using the VxWorks OS. Additional VME trigger delay module channels are needed to gate the data acquisition.

LASER

The electron bunch is initiated at the photocathode in the superconducting gun by a 9.38 MHz laser. The laser is a mode-locked Nd:YVO_4 master-oscillator power amplifier of custom design (Lumera Laser GmbH), required to achieve the low repetition rate and phase-locking requirements. The frequency-doubled output will be further conditioned by transverse and longitudinal shaping techniques [6]. Basic controls provided by the manufacturer are accessed using an attached PC running the Windows XP operating system. Efforts to implement a remote interface for the laser on the Controls system have previously involved the creation of a C++ server process in Cygwin that allows for direct integration with our infrastructure over Ethernet. Increasing concerns over Cyber Security issues and the hardware limitations of the included PC may result in a change in control strategy. The primary alternative would be to create a remote ADO server running on a LINUX machine that would interact with the laser PC over a serial connection.

Laser beam transport steering and shaping is achieved using a series of Newport gimbal mirror assemblies that can be moved by piezo actuators interfaced with a Newport 8-axis motor controller module. LINUX software has been developed to remotely send commands and return readings from each device.

An autocorrelation measurement system is under development, which will utilize an externally triggered fast ADC VME module in order to sample light from a photodiode synchronously with the laser pulses. Software has been developed to control another Newport linear actuator, which will be adapted to gather the digitized beam signal in conjunction with the motion of the photodiode.

Since the laser clock will provide the master timing for other systems at the ERL, a method for synthesizing related timing signals is needed. To that end, a Stanford Research Systems DG645 Digital Delay Generator will be used. LINUX software has been created to provide remote I/O over Ethernet for the device.

RF

Low-Level RF control of the 5-cell cavity tuner was first achieved using an analog phase-locked loop (PLL). This included motion controller and ADC VME boards for the cavity tuning feedback control, and a digital I/O VME board for gain control. Such a system was used successfully in support of the CET, and for further cavity tests since then. This included microphonics measurements [1] gathered from a fast ADC VME module that samples the LLRF signals in addition to the RF incident, reflected, and transmitted power. Over the course of multiple cold tests of the cavity, it became apparent that we were experiencing frequent gaps in data gathering from the digitizer when the total number of clients using the data exceeded a relatively small number. This issue was eventually resolved by first lowering the sampling rate from 16384 Hz to 4096 Hz, and then by configuring a data reflection server that shifts the client load from the FEC to the server's host machine. Details of a digital replacement for the analog LLRF platform can be found in ref [5, 7].

Additional CW RF power measurements are performed by two meters from Boonton. Data is relayed to the Controls System over a GPIB interface and GPIB to Ethernet bridge to a LINUX server process. Since there is also a need for pulsed RF power measurements, there is a separate LINUX program that acquires pulsed measurements from another Boonton meter over Ethernet.

POWER SUPPLIES

ERL main and trim dipole, quad, and solenoid magnets will be operated using power supplies from four different manufacturers: IE Power, BiRa, Kepco, and Danfysik. The controls for the former two types will leverage existing hardware and software interfaces, including the BNL Power Supply Controller (PSC) and commercial analog I/O VME modules. The Kepco and Danfysik units make use of serial I/O, though they will be configured for RS232 and RS485 respectively. User interfaces are provided through separate LINUX processes that utilize the Ethernet network and a serial Terminal Server.

Software development has been completed for the Kepco power supplies, and efforts to configure the intermediate hardware for the Danfysik interface are underway.

USER INTERFACE

The foundation for ERL user interfaces is firmly rooted in the existing application paradigm at the BNL C-AD: basic device interaction is performed through the Parameter Editing Tool (PET), live data graphing is available through the General Purpose Monitor (GPM) application, and logged data can be plotted through the LogView application.

Being that the ERL is a much smaller facility than others at the C-AD, we chose not to implement an alarm system. Instead, facility users have expressed a desire for synoptic displays that would graphically highlight any abnormal conditions, and allow the user to quickly drill down to the necessary subsystem controls in order to address the cause. Limited testing has been performed using MEDM on LINUX. We are currently planning on creating control pages using MEDM for the G5 tests, though evaluations of alternative graphical process control systems will likely continue in parallel.

REFERENCES

[1] B. Sheehy et al., "BNL 703 MHz Superconducting RF Cavity Testing", PAC'11, New York, March 2011, TUP056; http://www.JACoW.org.

[2] D. Kayran et al., "Status of High Current R&D Energy Recovery LINAC at Brookhaven National Laboratory", PAC'11, New York, March 2011, THP006: http://www.JACoW.org.

[3] Z. Altinbas et al., "The Machine Protection System for the R&D Energy Recovery LINAC", these proceedings

[4] R.H. Olsen et al., "Using An HDC/DCAM Camera for Beam Display and Analysis", ICALEPCS'09, Kobe, October 2009, WEP033, p. 474 (2009); http://www.JACoW.org.

[5] K.S. Smith et al., "Concept and Architecture of the RHIC LLRF Upgrade Platform", PAC'11, New York, March 2011, WEOBN5; http://www.JACoW.org.

[6] A.K. Sharma et al., Phys. Rev. ST Accel. Beams **12**, 033501 (2009).

[7] K.S. Smith et al., "Commissioning Results from the Recently Upgraded RHIC LLRF System", PAC'11, New York, March 2011, MOP298; http://www.JACoW.org.

CONTROL SYSTEM FOR LINEAR INDUCTION ACCELERATOR LIA-2: THE STRUCTURE AND HARDWARE

G. Fatkin*, P.A. Bak, A.M. Batrakov, P.V. Logachev, A. Panov, A.V. Pavlenko, V.Ya. Sazansky
Budker Institute of Nuclear Physics, Novosibirsk

Abstract

Power Linear Induction Accelerator (LIA) for flash radiography is commissioned in Budker Institute of Nuclear Physics (BINP) in Novosibirsk. It is a facility producing pulsed electron beam with energy 2 MeV, current 1 kA and spot size less than 2 mm. Beam quality and reliability of facility are required for radiography experiments. Features and structure of distributed control system ensuring these demands are discussed. Control system hardware based on CompactPCI and PMC standards is embedded directly into power pulsed generators. CAN-BUS and Ethernet are used as interconnection protocols. Parameters and essential details for measuring equipment and control electronics produced in BINP and available COTS are presented. The first results of the control system commissioning, reliability and hardware vitality are discussed.

INTRODUCTION

Flash X-Ray radiography facilities are used for registration in hydrodynamic experiments. Linear Induction Accelerators (LIA's) are considered to be the most prominent source of electrons which are converted to gamma radiation [1]. The have because of their relatively low emittance, high currents(up to a few kA) and energy (up to tens MeV).

LIA-2 was originally designed as an injector for 20 MeV linear induction accelerator, but it can be used as an independent 2 MeV X-ray source for flash radiography with high space resolution. This machine has been built with the goal to keep the minimum possible emittace and therefore the minimum beam spot size on the target. The reliablity of the machine is of great importance. Basic parameters of LIA-2 are presented in Table 1.

Table 1: LIA-2 Basic Parameters

Parameter (Units)	Value
Maximum electron beam energy (MeV)	2.0
Maximum electron beam current (kA)	2.0
Number of pulses	2
Cathode heater DC power (kW)	2.5
Time interval between pulses (μ s)	2-10
Pulse duration, flat top $\pm 4\%$ (ns)	200
Maximum repetition rate (Hz)	0.1
Min. beam spot size FWHM on the target (mm)	1.5

* george.fatkin@gmail.com

Let us note, that there are only a few linear induction accelerators for flash radiography around the world. Well known facilities are: FXR (Livermore Laboratory, USA), AIRIX (PEM, France), DARHT (Los-Alamos, USA), DRAGON (China) [2], [3], [4], [5]. The control system of DARHT-2 is described in [6]. The approach that allows us to get the minimum possible emittance leads to a large amount of controlled devices (for comparison DARHT-2 has 74 induction cells for 20 MeV installation, we have 48 induction cells for 2 MeV).

INSTALLATION STRUCTURE

The accelerating stucture (see fig. 1) includes 1 MV, 2kA diode whith dispenser cathode and 1 MV accelerating section placed just after the diode exit. Both sections of high-voltage pulsed transformer include linear chain of 96 induction cells put on accelerating tube.

The voltage for supplying inductors is generated by 48 identical modulators: pulsed power devices composed of pulse forming network, high-voltage switch (thyratron), demagnetizing generator and control electronics. Output pulse voltage is 50 kV, duration – 300 ns, output current comes up to 5 kA. Each modulator feeds two induction cells.

Low-temperature dispenser thermo cathode is heated up by a special 70 A generator. Cathode heating current should be turned off at the moment of shot to ensure zero magnetic interferention. Four short focusing solenoids are powered by pulsed current generators which should be synchronized. Three dipole correctors are powered by direct current. Beam diagnostic system consists of two beam current transformers, capacitor-based pulsed voltage dividers, X-Y strip-line beam position monitor (BPM) and movable Faraday cup.

Normal operating cycle of LIA consists of two stages: slow and fast. During slow stage the elements of installation are prepared for fast operations: forming networks and demagnetizing generators are charged, thyratrons are preset, the states of various elements are checked. The duration of slow stage is 20-30 ms. If all elements of LIA are in normal condition, then the fast stage is initiated. This stage starts from the moment of inductor's demagnetizing and finishes with beam generation. Duration of this stage is about 150 μs. If any failures occur during this stage, interlock subsystem disables the experiment start.

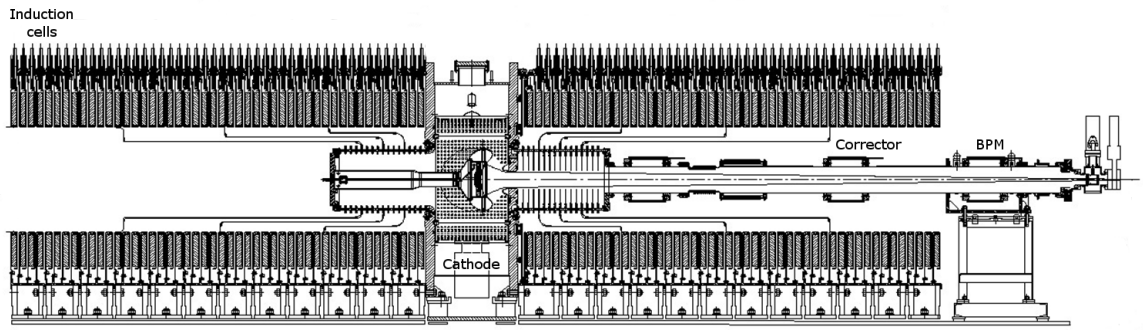


Figure 1: LIA accelerating structure.

CONTROL SYSTEM FEATURES

Let us formulate the features of control system of LIA. The hardware synchronization system is needed to provide trigger pulses to modulators during both stages of operation. The interlock system is an absolute necessity because of high experiment cost and preparation time. To ensure beam quality and to provide full information on installation performance more than 200 waveforms per shot should be recorded.

Planned time of operation of LIA is tens of years. Thus the control system must be based on modern widespread hardware standards and software.

Most of the controlled devices are high-voltage pulse generators. Special precautions should be taken to protect control electronics from high-voltage breakdowns. Electronics and transmission lines should also be protected against interference induced by high voltage pulses.

Control system must be scalable to allow adding induction cells. The most numerous control object is modulator. They are grouped in sections by 6 in each and linearly distributed along accelerator's length. It is convenient to have control elements structured accordingly.

Therefore we can formulate following features of the control system of LIA:

- Linearly structured
- Strict hardware synchronization
- Hardware interlock subsystem
- Adequate waveform recording subsystem
- High-power nature of the installation considered
- Based on modern widespread standards
- Scalable

STRUCTURE

The control system of LIA is functionally divided into four subsystems: pulsed power control subsystem, timing subsystem, waveform recording subsystem and interlock subsystem.

Pulsed power control subsystem is controlling all devices during slow phase of operation. This is mainly done by software using CAN-bus.

The timing subsystem generates and distributes trigger pulses to modulators and other devices. Each modulator needs 6 trigger pulses for proper operation. The delay of pulses initiating discharge should be individually adjusted for each thyratron with accuracy less than 10 ns. Other pulses may be set with accuracy about 100 ns. Altogether there are 288 timing channels.

The waveform recording subsystem provides information about installation state and beam quality. Flash radiography facilities are used to conduct very expensive experiments. To estimate the reliability of the installation, 96 "fast" signals (duration about 300-1000 ns) are recorded on each shot. Recorded data is stored and statistically analyzed. Overall accuracy around 1% for these signals is needed to ensure beam quality. In addition about one hundred technological signals are recorded and analyzed during the slow phase of operation.

The interlock subsystem prevents damage to the installation and prohibits experiment start if alarm signal from any of the devices is recieved. It has to be absolutely reliable.

The structure of the control system of injector is shown on fig. 2. High-voltage system for injector consists of 8 identical sections. Each section contains 6 modulators supplying 12 induction cells. Local controller performs all operations necessary for the functioning of section: modulator timing, waveform recording, interlock functionality, CAN-bus interaction. Local controller is 6U 8-slot CPCI crate installed directly at the modulators section rack. This is done to minimize cables length thus minimizing noise signals. Local controller is shown on fig. 3.

Six modulators are connected to common CAN bus, which is used for their setup and controls slow changing parameters. Trigger pulses for demagnetizing and thyratron preparation are received from central crate and split to all modulators using the F-16 splitter board. All other fast thyratron pulses are generated using digital delay line which is started by a trigger pulse from main controller.

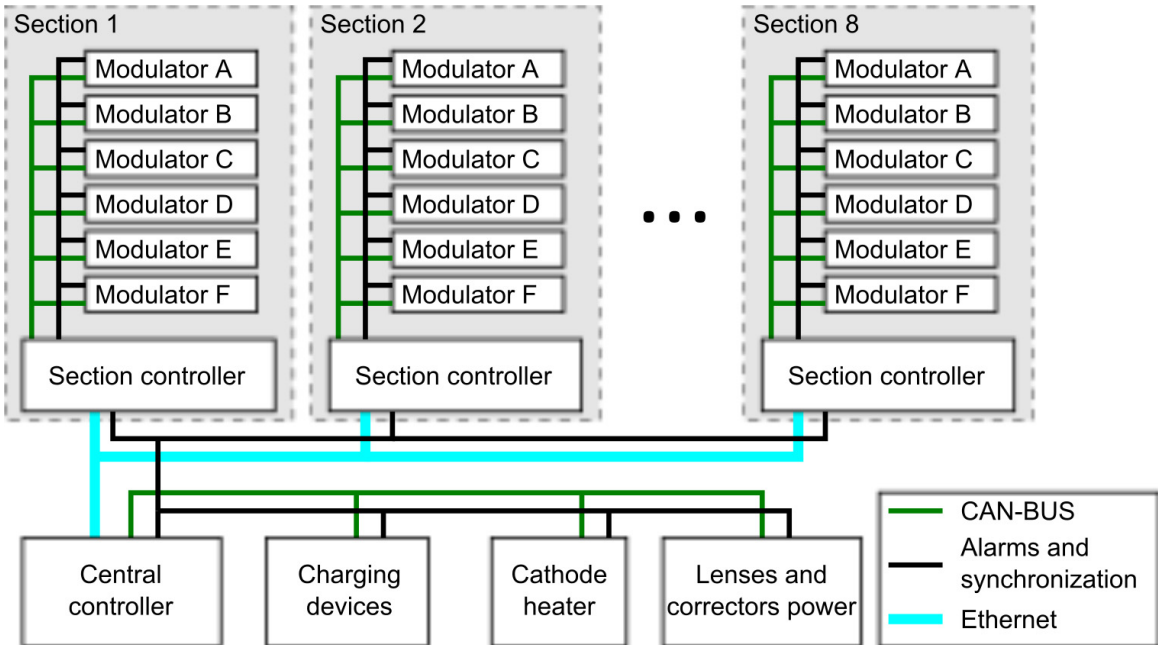


Figure 2: The structure of LIA control system.

Local controller also gathers alarms from all modulators and generates section alarm signal.

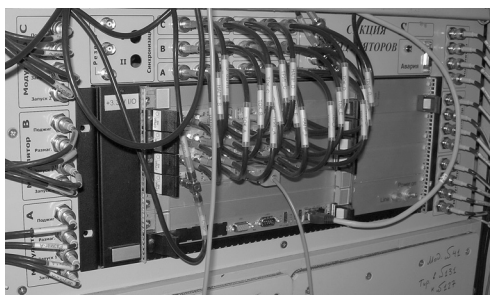


Figure 3: Local controller.

The central controller synchronizes and supervises local controllers operation. It controls charging devices, lenses, BPM and cathode heater via CAN-bus. It also gathers alarm signals from sections and raises general alarm in case of breakdown in any section. All controllers and operator's PC are interconnected using Ethernet.

The hardware synchronization of the installation is done using general following pulses: local controller starts, demagnetizing and thyrathron preparation signals, cathode heater turn off, lenses starts, experiment start. This pulses are transferred using coaxial lines.

Hardware

Common mezzanine card (PMC) and CompactPCI standards were chosen as a hardware basis for control system. CompactPCI was chosen as a widespread industrial standard compatible with x86 architecture. It allows to use widespread operating systems (Windows, Linux) and benefits from wide range of COTS automation equipment. PMC cards allow interchanging host system (CompactPCI, VME, PCI,...), thus electronics made for the LIA control system could be used for other tasks. CompactPCI rear input output (RIO) modules are used to effectively cope with huge amount of cables.

LIA is high current and high-voltage facility with operation cycle in nanosecond time region. Therefore survivability of electronics at the environment of periodic high-voltage discharges is an issue. We decided to develop hardware directly interfacing to high-voltage devices ourselves. Doing this we have the possibility to improve the survivability of hardware if needed. This approach proved to be useful to overcome false shots due to imperfect grounding in synchronization subsystem. We have unified hardware of all controllers to make maintenance easier.

Four modules were specially developed in BINP for use in LIA control system. Waveform recorders ADC200-ME and ADC812-ME were developed for waveform recording sub-system. Both of them are in PMC standard. Their specifications are given in table 2. ADC200-ME is used for the measurement of fast signals: inductor currents and voltages, BPM signal, signals form faraday cup and current transformers. ADC812-ME is used for slow waveforms recording. Inputs of two ADC812-ME's are placed on CPCI RIO module that is also used to route device alarms to F-16.

Two other modules were developed for interlock and synchronization sub-systems: DL200-ME (digital delay line with built-in interlocks) and F-16 driver/splitter. DL200-ME specifications are shown in table 3. Interlock channels are optically decoupled to protect the board from high-voltage breakdowns. FPGA is used to allow a pro-

Table 2: Waveform Recorders Specifications

	ADC200-ME	ADC812-ME
Channels	2 simultaneous	8 simultaneous
Conversion rate	200 MSPS	5 MSPS
Accuracy	12 bit	12 bit
Memory	1M×2×12 bit	1M ×2×12 bit

grammable interlock logic. There are two modifications of DL200-ME firmware, fast and slow. Fast modification (DL200-ME-F) is used in local controllers. Slow modification (DL200-ME-S) is used in main controller, provides 80 ns step and has specific interlock logic.

Table 3: DL200-ME Digital Delay Line Specifications

Output channels	16
Step	5ns (80 ns for S modification)
Jitter	<0.5 ns
Counter	15 bit (23 bit for S modification)
Interlock channels	16
Alarm channels	2

F-16 is a RIO CPCI module. It has two modifications: driver and splitter. F-16 driver is controlled by DL-200ME. It forms 16 decoupled synchronizing signals and gathers alarm signals. Another modification is F-16 splitter that splits two incoming signals on 8 outputs each. It is used to split slow synchronization pulses from main controller to all induction drivers in section.

Kontron [7] CP-6000 processor boards are used as central processing units in all crates. They have 2 Ethernet ports (4 with backplane) and PMC mezzanine slot. 4GB CompactFlash memory contains OS and control software.

Two boards are used as mezzanine carriers for PMC modules: Kontron CP-690HS and a module with built-in CAN controller constructed in BINP. TEWS [8] TPMC810-10 is used as a mezzannine CAN controller with two DB-9 connectors.

CONCLUSION

Features of LIA control system were formulated and the structure of the control system was designed. CompactPCI and PMC standards are used for control system hardware. Hardware for the control system has been developed, manufactured and tested in operation. Some of this hardware is used at other works in BINP.

LIA-2 was tested at single pulse regime of operation. Diamaeter of the electron beam diameter on the target (FWHM) less than 1.3 mm was achieved. Further experiments are continued on the customer site. High survivability of hardware was demonstrated in the environment of high-voltage discharges. The experience gained during creation and operation of LIA is usable for design of the control system of 20-MeV flash X-Ray radiography complex.

REFERENCES

[1] C. Ecdahl, “Modern electron accelerators for radiography”, 2001 IEEE Pulsed Power Conference, Las Vegas, Nevada, 2001, plenary presentation

[2] M. Ong, et. al., “LLNL Flash X-Ray Radiography Machine (FXR) Double-Pulse Upgrade Diagnostics”. 1997 IEEE Pulsed Power Conference, 1997, pp. 430-436

[3] M Mouillet, et. al., “First Results of the AIRIX Induction Accelerator”, XX International Linac Conference, Monterey, California, 2000, pp. 491-493

[4] M.J. Burns, et. al., “Status of the DARHT Phase 2 Long-Pulse Accelerator”, Particle Accelerator Conference 2001, Chicago, IL, 2001, pp. 325-330

[5] J.J. Deng, et. al., “Design of the Dragon-I Linear Induction Accelerator”, LINAC 2002, Gyeongju, Korea, pp. 40-42

[6] R. Archuleta, L. Sanchez, “The DARHT Data Acquisition, Archival, Analysis and Instrument Control System (DAAAC), And Network Infrastructure.”, XXIV Linear Accelerator Conference, Victoria, BC, Canada, 2008, pp. 337-339

[7] http://emea.kontron.com/products

[8] http://www.tews.com

AN EPICS IOC BUILDER

M.G. Abbott, T. Cobb, Diamond Light Source, Oxfordshire, UK

Abstract

An EPICS IO Controller (IOC) is typically assembled from a number of standard components each with potentially quite complex hardware or software initialisation procedures intermixed with a good deal of repetitive boilerplate code. Assembling and maintaining a complex IOC can be a quite difficult and error prone process, particularly if the components are unfamiliar. The EPICS IOC builder is a Python library designed to automate the assembly of a complete IOC from a concise component level description. The dependencies and interactions between components as well as their detailed initialisation procedures are automatically managed by the IOC builder through component description files maintained with the individual components. At Diamond Light Source we have a large library of components that can be assembled into EPICS IOCs. The IOC Builder is further finding increasing use in helping non-expert users to assemble an IOC without specialist knowledge.

INTRODUCTION

The IOC builder is a Python framework designed to automate the process of assembling complex IOCs from existing components. It also provides facilities for generating databases, either as auxiliary records to link components in an IOC, or for the generation of standalone templates.

At Diamond the IOC builder has been used to generate Diagnostics and Power Supply controller IOCs, which show a combination of repetition and variation which is perfect for scripted generation, most area detector IOCs which need complex setting up, and a number of other photon beamline IOCs. The builder has also been used to generate the templates for the Libera EPICS driver [1, 2].

An IOC is normally assembled from existing EPICS support modules together with a small amount of IOC and module specific glue. Input to the IOC builder is either in the form of a structured Python script or a list of component instances specified in XML. The XML version of the IOC builder is designed for more routine IOC definitions where all that really needs to be specified is a list of components provided by EPICS modules and their parameters.

Each EPICS support module at Diamond can be found in a well defined location and has a clearly defined structure. As part of this structure IOC builder definitions are defined by each support module; these define the components such as templates or hardware drivers that go into an IOC.

```
import iocbuilder
iocbuilder.ConfigureIOC()
from iocbuilder import *

# Load support module definitions for needed modules
ModuleVersion('ipac',         '2-8dls4-5')
ModuleVersion('Hy8515',       '3-9')
ModuleVersion('asyn',         '4-10')
ModuleVersion('streamDevice', '2-4dls2-1')
ModuleVersion('newstep',      '1-4')

# Define the hardware resources used
card4 = modules.ipac.Hy8002(4)
serial = card4.Hy8515(0)

# Create two newstep controllers
for ch in range(2):
    asyn = modules.asyn.AsynSerial(serial.channel(ch))
    modules.newstep.NSC200(
        M = 'TS-TEST-DEV-01', P = '',
        PORT = asyn.DeviceName(), CH = ch)

WriteNamedIoc('ioc', 'TEST-IOC')
```

Figure 1: Complete script to assemble a simple IOC with serial hardware and stream device protocol definitions.

BUILDING AN EXAMPLE IOC

Figure 1 shows a complete IOC builder script for assembling a simple IOC, which in this case uses the streamDevice support module to communicate over a serial port provided by the Hytec 8515 IP card. Here we need five support modules with specific versions loaded by the calls to `ModuleVersion` and two items of hardware (the 8515 serial IP module and a carrier card in which to install it), and we create two instances of the newstep NSC200 component before writing out the entire IOC to the `ioc` subdirectory.

The IOC that is written out contains a top level make file in standard EPICS style together with a standard `configure` directory containing a file `RELEASE` with the relevant `ModuleVersion` definitions converted into EPICS version definitions. The rest of the IOC directory contains the necessary makefiles to link all the required libraries and EPICS dbd files together and template substitutions to complete the definitions.

Note that inter-module dependencies are automatically handled, the only thing the user needs to do here is to specify module versions in the correct order. The XML builder even automates this step by taking as input an existing `configure/RELEASE` file and walking the associated dependency tree.

USING THE IOC BUILDER

An IOC build script normally has a fairly stereotypical structure, consisting of four stages:

- Initialisation and configuration.
- Loading module definitions.
- Creation of IOC resources.
- Generation of output.

Initialisation and Configuration. The IOC builder can be used to generate IOCs to target a variety of architectures; at Diamond we support VxWorks, Linux and Windows. Alternatively the builder can be used to just generate a record template file. To choose between these options the builder must be configured before importing most of the iocbuilder symbols by calling one of the functions `ConfigureIOC` or `ConfigureTemplate`.

Loading Module Definitions. Each EPICS support module at Diamond contains IOC builder definitions specific to that module in the file `etc/builder.py`, and these definitions must be loaded into the builder by calling the builder `ModuleVersion` function before the support module can be used. Many support modules have dependencies on other support modules, so these dependencies must be loaded in the correct sequence.

Creation of IOC Resources. At its simplest an IOC can be defined as just a list of component instances where each component is a resource defined by a support module. The XML builder uses this approach where the input to the builder is essentially a table of component names and their arguments.

More generally it is sensible to structure an IOC definition script with a little bit of care. Hardware definitions can be placed first and it makes sense to separate them from the software resources that use them. Any required resource initialisation will automatically be created as required.

All the resources defined by individual support modules are loaded as Python submodules of the module `iocbuilder.modules` as can be seen in Figure 1.

Generation of Output. Typically the IOC builder can either write out a complete Diamond conformant IOC directory structure, or a single template file, using the functions `WriteNamedIoc` or `WriteRecords` as appropriate.

Creating EPICS Record Templates

The IOC builder was originally written to create complex databases, as gluing together hand-written templates with substitutions or macro processing rapidly becomes cumbersome. This functionality is available for normal IOC generation when some EPICS records are required to connect components together, but is more useful for building complex EPICS record templates.

The basic building block for doing this is the `iocbuilder.records` object which has record constructing methods for each record type. This object is automatically populated as DBD files are loaded, both from EPICS base on startup and from individual module definitions. Records are constructed in memory as Python objects with attributes for each possible record field, and assignments to these fields are automatically checked against the DBD definitions.

To simplify the naming of records when creating them a configurable record naming convention is established when initialising the IOC builder, both enforcing the Diamond record naming convention, and allowing the prefix of the name to be established separately.

Frequently records are generated in association with hardware with specific values written to the `DTYP` and `INP` or `OUT` fields. The builder supports this with record constructors tied to hardware instances which automatically populate these fields.

Figure 2 shows a fragment of a script using these mechanisms to generate a database with 22 record and 217 field definitions.

```
SetChannelName('SA')
power = Libera.ai('POWER',
    LOPR=-80, HOPR=10, ESLO=1e-6, EGU='dBm', PREC=3,
    DESC = 'Absolute input power')
current = Libera.ai('CURRENT',
    LOPR=0, HOPR=500, ESLO=1e-5, EGU='mA', PREC=3,
    DESC = 'SA input current')
Trigger(False, ABCD_() + ABCD_N() + XYQS_(4) +
    [power, current] + MaxAdc())
UnsetChannelName()
```

Figure 2: Fragment of script to generate an EPICS template. Here `Libera` is a device instance, `Trigger` creates a fanout record, and its arguments are functions for generating further records.

Using the IOC builder with an XML file

While the Python interface is useful for building many similar IOCs, many of our IOCs take the form of hundreds of similar objects with slightly different parameters, e.g. motors with different resolutions. This can be better presented in a tabular form, so the IOC builder contains a utility called the XML builder that can build an IOC from parameters stored in an XML file and an interactive graphical editor `xeb` ("XML Editor for the Builder", see Figure 3) that can read and write the correct format XML file. To support this, each argument of every builder object must be annotated with a description and a type as described in the `ArgInfo` section below. The editor can then do suitable error checking and type casting of the arguments, and present descriptive prompts to the user.

In fact, the XML builder and Python scripting are two equivalent approaches with similar power that suit different working styles.

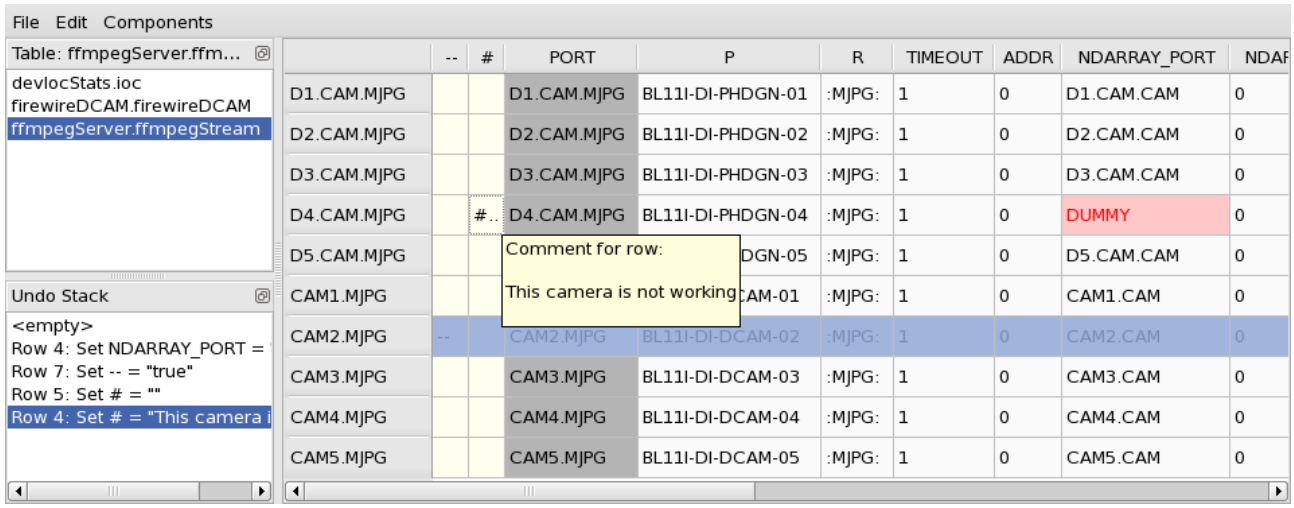


Figure 3: XEB: XML Builder Editor showing camera configuration for an area detector IOC.

MODULE DEFINITIONS

For an EPICS support module to be useful with the IOC builder it is necessary to create IOC builder class definitions for the resources defined by the module. These should encapsulate any module, linkage and DBD dependencies, and should also define any startup script commands needed by the support module. Ideally a good set of builder definitions should capture all the specific knowledge required to use the support module so that the developer of the IOC doesn't need to know.

The builder definitions for a support module are loaded from the file `etc/builder.py` in the base of the support module, which is loaded as a submodule of `iocbuilder.modules`. Builder resources are generally defined by creating subclasses of the `ModuleBase` class exported by the builder, largely as subclasses of `Substitution` to declare EPICS database templates, or of `Device` to declare resources with linkage or initialisation dependencies.

A support module builder definition file should open with imports from `iocbuilder.modules` of all support modules on which it depends to ensure that all dependencies have been loaded.

Module Base

Component resources are normally declared as subclasses of `ModuleBase`. This class performs a number of important functions:

- Ensures the module is included in the build.
- Ensures any module dependencies are resolved, this includes ensuring that dependent modules are linked and initialised in the correct order, and may involve automatically loading other modules.
- Defines path dependent access to files defined by the module. For example, this is used to locate the template file in a `Substitution` definition.

Any subclass of `ModuleBase` can define the value `Dependencies` to specify a list of other modules which must be instantiated before this class is used.

```
class NSC200a(Substitution):
    Arguments = ['P', 'M', 'CH', 'PORT']
    TemplateFile = 'NSC200.template'

    ArgInfo = makeArgInfo(
        P    = Simple('Device Prefix', str),
        M    = Simple('Device Suffix', str),
        CH   = Simple('Channel number', int),
        PORT = Simple('Asyn port string', str))

class NSC200b(AutoSubstitution):
    TemplateFile = 'NSC200.template'
```

Figure 4: Template definition examples showing both manual and automatic template definitions.

Template Definitions

Template definitions are the simplest to write, and are defined as subclasses of the `Substitution` builder class. The defined subclass must define the template file name, an enumeration of the arguments required by the template, and their descriptions. The definition of `NSC200a` in Figure 4 shows this.

The constructor for `NSC200a` will then expect to be called with exactly the listed keyword arguments and will fail otherwise. An alternative constructor can be defined for the class if appropriate.

The class `AutoSubstitution` is a subclass of `Substitution` which automatically searches the specified template file for template parameters to populate the `Arguments` and `ArgInfo` fields, and so allows a template definition to be just two lines as show in the definition of `NSC200b` in Figure 4; this is equivalent to `NSC200a`.

Every instance of a `Substitution` subclass results in the appropriate entries in the `Db/Makefile` and an associated `.substitutions` file.

Device Definitions

The `Device` class is used for component resources which need any of the following elements:

- Startup script initialisation.
- Loading of DBD files, either for new record definitions or for record device type definitions.
- Loading of libraries.
- Custom entries in Makefiles.

A component resource is created by defining a subclass of `Device` and setting values for a number of values, including:

LibFileList A list of libraries that must be linked when using this component.

DbdFileList A list of DBD files defining the EPICS resources provided by and making up this component.

Initialise() This method will be called during the generation of the IOC startup script to generate any device specific initialisation code.

Typically every support module should have a `Device` definition which defines `LibFileList` and `DbdFileList` to record the libraries and DBD files used to load the support module and which is marked as a dependency of the other components of the support module.

Component Argument Descriptions

The XML builder expects every builder component to be annotated with "metadata" documenting the parameters required to instantiate that component. This is done by creating an `ArgInfo` object, for example as shown in the definition of `NSC200a` in Figure 4.

`makeArgInfo` optionally takes the `__init__` method as the first argument, if a custom `__init__` method has been defined, followed by named arguments describing each argument that should be passed to `__init__`. Each of these arguments can be one of:

Simple A simple type

Ident An identifier, lets you specify that this argument should be something of a particular type

Choice One of a list

Enum As choice, but pass the index of the selection to the `__init__` method

It is also possible to add ArgInfo objects together, and filter them using the `filtered` method. This allows more complicated argument structures to be built up.

The `AutoSubstitution` simplifies this process for templates: as the template file is scanned for arguments, an `ArgInfo` object is automatically assembled at the same time. Specially formatted comments in the template file can be used to provide argument descriptions.

```
class Hy8002(IpCarrier):
    MaxIpSlots = 4      # 4 IP slots in this card

    def __init__(self, slot, intLevel=2):
        self.__super.__init__(slot)
        self.intLevel = intLevel

    ArgInfo = makeArgInfo(__init__,
        slot     = Simple('VME Slot number', int),
        intLevel = Simple('VME Interrupt Level', int))

    @classmethod
    def UseModule(cls):     # Assign shared interrupt
        super(Hy8002, cls).UseModule()
        cls.swapint = cls.AllocateIntVector()

    def Initialise(self):
        self.InitialiseCarrier(
            'EXTHy8002', self.slot, self.intLevel,
            self.swapint)
```

Figure 5: Module definition for Hy8002 IP carrier card.

An Example Device Definition

Figure 5 shows the complete module definition for the Hy8002 IP carrier device, part of the ipac support module. The `IpCarrier` class is a subclass of `Device` providing special support for IP carrier cards.

This device is fairly unusual in having only two arguments, but typical in the use of defaults. The `UseModule` method is called by `ModuleBase` immediately before creating the first instance of this class.

CONCLUSIONS

The IOC builder is being used for an increasing number of IOCs at Diamond and has been under continuous development for around five years by both authors.

There are some obstacles to using the IOC builder outside Diamond, the largest being the dependency on the Diamond directory structure. The builder code is reasonably well structured, though a transition to documentation using Doxygen seems to have been unhelpful.

Work on the builder continues in response to internal Diamond developments.

This work was first presented outside Diamond at the EPICS workshop at ICALEPCS 2009 [3].

REFERENCES

[1] M.G. Abbott, G. Rehm, I.S. Uzun, "The Diamond Light Source Control System Interface to the Libera Electron Beam Position Monitors", ICALEPCS 2009.

[2] `http://controls.diamond.ac.uk/downloads/other/libera`

[3] M.G. Abbott, "EPICS IOC builder", EPICS collaboration meeting, ICALEPCS 2009, `http://kds.kek.jp/contributionDisplay.py?contribId=16&confId=3834`.

CONTROLVIEW TO EPICS CONVERSION OF THE TRIUMF TR13 CYCLOTRON CONTROL SYSTEM

D.B. Morris, TRIUMF, Vancouver, Canada.

Abstract

The TRIUMF TR13 Cyclotron Control System was developed in 1993 using Allen Bradley PLCs and ControlView. A console replacement project using the EPICS toolkit was started in Fall 2009 with the strict requirement that the PLC code not be modified. Access to the operating machine would be limited due to production schedules. A complete mock-up of the PLC control system was built to allow parallel development and testing without interfering with the production system. The deployment allows both systems to operate simultaneously, easing verification of all functions. A major modification was required to the EPICS Allen Bradley PLC5 Device Support software to support the original PLC programming schema. EDM screens were manually built to create displays similar to the original ControlView screens, thus reducing operator re-training. A discussion is presented on some of the problems encountered and their solutions.

INTRODUCTION

The TRIUMF TR13 Cyclotron was designed by TRIUMF and built in collaboration with EBCO Technologies, now ACSI [1][2]. The control system was built by TRIUMF using two Allen Bradley PLC5 Programmable Logic Controllers (PLC) [3] and the ControlView OPI toolkit supported by the Chronos [4] multitasking system which runs on top of Microsoft DOS 6.22. The operator console computers running the ControlView displays are Intel 486 based PCs. They communicate with the PLCs using the Allen Bradley DataHighway Plus (DH+), a serial highway operating at 57.6 kbaud. Figure 1 shows a block diagram of the original system, which has been running essentially unmodified since 1993.

UPGRADE RATIONALE

The upgrade was driven by several factors:

- The ControlView toolkit is obsolete and the ageing hardware is difficult to support. Newer hardware is not supported by ControlView.
- The system performance degraded with a second installed console due to limitations of the DH+ serial highway.
- The dual console system was not supported by ControlView. Avoidance of conflicts between the two control masters was procedural and error-prone.
- The consoles are located in a harsh environment with high noise levels and wide temperature swings. Controlview does not allow remote consoles, so the operations area could not be relocated or duplicated, remotely.

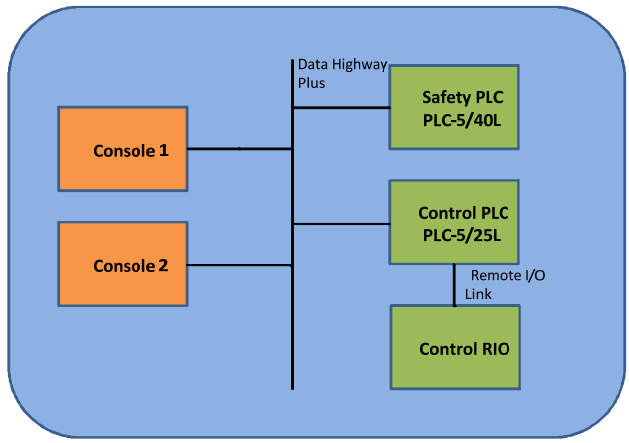


Figure 1: Original System.

UPGRADE LIMITATIONS

The principal limitation for the upgrade was that the PLC program must not be modified as this would trigger a re-licensing requirement of the entire cyclotron. In addition, the upgrade was not to affect the production runs which are scheduled every weekday, varying in length and frequency during the day. There is no regular pattern which would allow testing periods.

REPLACEMENT OPTIONS

On the software side, it was decided from the outset to use the EPICS control system tool-kit which is used for the ISAC control system. Alternative options, such as using the Allen Bradley RSView product or other commercial OPI products were not considered because of cost and in order to conserve the control group resources.

On the hardware side, several options were investigated for the replacement and finally rejected:

- Installing an Ethernet Coprocessor into the PLC chassis because it would have required PLC module re-addressing and PLC program changes.
- Use of a Serial to DH+ module because of speed limitations.
- Replacing the PLC CPU with a ControlLogix or other PLC because it would require rewriting the PLC program.
- Installing a Datalink translator between ModBus/TCP and DataHighway Plus because of speed limitations and possible size limits of the translation tables.

It was decided to replace the PLC CPUs with fully compatible PLC-5 Ethernet Processors. This would require no PLC program changes and allow deployment

of an EPICS OPI with fast communication between console and PLC. The new system could be deployed in parallel with the ControlView system greatly simplifying commissioning. Figure 2 shows a block diagram of the upgraded system.

This solution required replacement of the PLC program development software, Ladder Logistics from ICOM, as it did not support the Ethernet enableed PLC family. An available copy of ICOM's WinLogic 5 was initially used to monitor the PLC while the EPICS interface was developed. WinLogic 5 was eventually replaced with a current version of Rockwell Automation RSLogix.

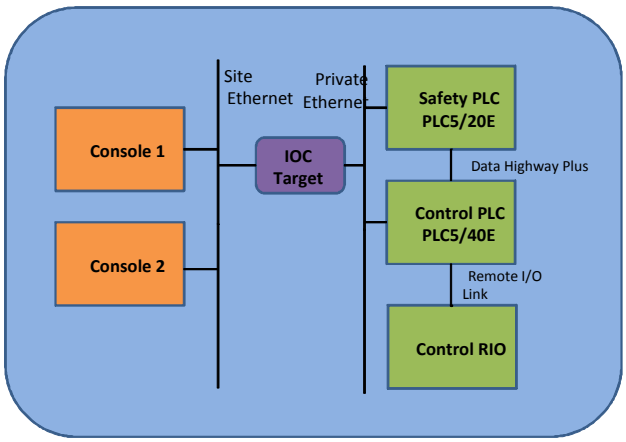


Figure 2: Replacement System.

FEASIBILITY TEST

A test system was set up to prove that the selected methodology was sound and that the underlying EPICS device support would work for the TR13. A mock-up was built of the Control and Safety PLC systems by scavenging parts and purchasing an assortment of modules from E-Bay vendors (see Fig. 3). The total investment in the mock-up was about $2500. With the hardware assembled, the existing PLC software was configured for the Ethernet PLCs and installed.

An EPICS IOC application running on a Linux based platform was built using the PLC5 Device Support library from SNS [5]. A rudimentary EPICS run-time database was created to access some of the PLC memory locations. The PLC status was monitored using WinLogic 5. Successful communication in both directions was observed between the IOC and PLC.

SYSTEM DEVELOPMENT

EPICS Run-time Databases

PLC addresses and device functionality were identified by examining the PLC ladder logic. Differences between the tag address layout in the TR13 PLC memory and ISAC conventions precluded the use of existing device and component schematics. Because of the small size and one-off nature of the system it was decided to instantiate the devices interactively into schematic diagrams for the different TR13 sections and not to upgrade available tools [6]. Following the methodology used for the ISAC control system, EPICS run-time databases were designed using the Capfast [7] schematic editor from which the run-time databases were generated.

OPI Screens

The TR13 operations group required that the EPICS display screens be as similar as possible to the existing ControlView displays in order to reduce operator re-training and improve acceptance of the new system. This required that all screens be manually created, since the auto-generation tools that were available [8] would have needed extensive modifications to handle the visual representation of control components and especially of the device interlocks. The effort to modify the ISAC tools was deemed too high given the one-off nature or the TR13 system.

Some 160 OPI screens were developed using edm, the standard graphical display tool for EPICS systems at TRIUMF.

The ControlView interface had been optimized to use keyboard input extensively, with Function keys, arrow keys and others triggering actions in the system. In contrast, edm is based on the X-Windows system, and provides less in the way of keyboard support. Although some edm widgets support arrow keys for increment and decrement, the standard coarse/fine adjustments available in the ControlView system could not be implemented with the same level convenience for the operators.

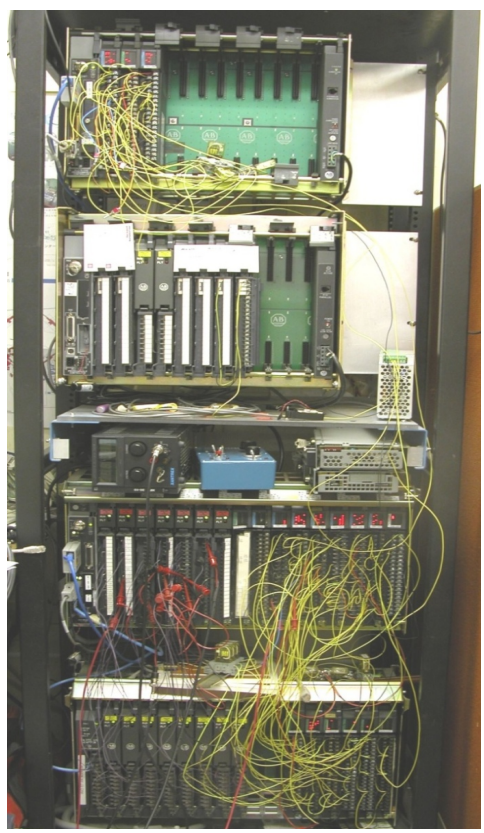


Figure 3: The mock-up system with wires simulating device feedback.

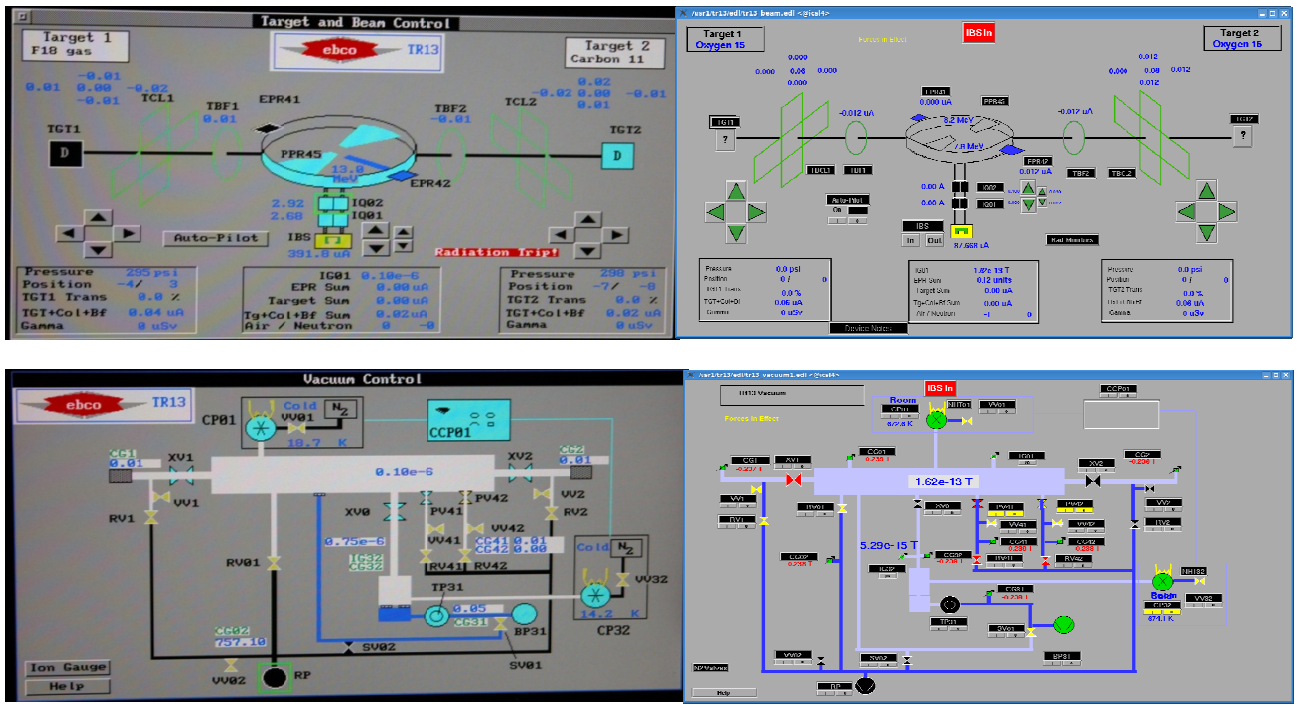


Figure 4: Examples of ControlView (left) and EDM (right) screens.

IMPLEMENTATION

In order to not affect the production schedule of the cyclotron, maintenance days were used to install and test the new system incrementally:

1. The Safety PLC was replaced with an Ethernet enabled unit, the Safety program was loaded and started, and the safety system was tested for proper operation.
2. The Control PLC was replaced and regular operation was tested.
3. The EPICS SoftIOC with a small database was started on the new console computer and operation of the ControlView system was monitored for anomalies.
4. Control was tested using the EPICS interface. During this phase some irregular system behaviour was observed, such as certain valves turning off unexpectedly.
   Investigation of the structure of the PLC5 Device Support code traced the fault to a design philosophy from SNS that was incompatible with the TR13. The SNS implementation uses a PLC memory block for efficient command writes to the PLC. The driver monitors this transfer block, and re-asserts any words that change in the block to ensure all commands are valid. In the TR13 PLC memory, there is no separation of command bits, status bits, internal bits and one-shot bits into different memory blocks. This led to unintended write operations with unexpected side effects by the SNS driver. These problems were resolved by the author with an extensive re-write of the PLC5 Device Support. Several new features were added, including direct bit and word write, optional scanning of bit and word changes, timer and counter accumulator reads and many new diagnostic tools at all levels of the driver. The driver was also modified to handle proper octal/decimal conversion, as the PLC programming tools use a mix of these for I/O bits and words versus internal memory. All the existing database addresses had to be modified to handle the new driver structure.
5. The new driver was installed in the IOC and further testing resulted in acceptance of the driver for operations.

IOC PLATFORM

As can be seen from Fig. 1 and Fig. 2, the EPICS implementation introduces an IOC between the operator consoles and the PLCs. This alleviates some of the problems encountered with console “competition” under ControlView. The IOC could – in principle – be implemented as a software process on one of the consoles. At TRIUMF this is never done because of the risks of user interference with the IOC application, not to mention the possibility of a user shutting the IOC application down.

Typical installations for the TRIUMF ISAC facility provide an IOC server that will run several SoftIOCs. In the case of the TR13, a dedicated target platform was required that could be placed in the cabinet with the PLC hardware.

An Intel based platform was preferred, as all our Linux development systems are Intel based, avoiding cross-

compiling with its associated infrastructure. To improve reliability there should be no moving parts.

An Intel Atom based Netbook was used as a test platform to test processor load and performance. Results showed sufficient power and resources to support the IOC processing approximately 2900 EPICS records. The final IOC was implemented on an Atom - Mini-ITX platform.

This solution proved efficient and cost-effective and was therefore adopted as a standard small IOC platform for EPICS at TRIUMF. It has the additional possibility of performing specialized hardware I/O via PCI cards.

ACCESS SECURITY

Several layers of security were implemented for this system to prevent unauthorized operation while still allowing operators to monitor the cyclotron remotely. The console computers limit access through host allow/deny rules and user identification. The IOC computer implements a firewall to restrict access to a select few site computers such as the consoles and the development computers. The firewall also blocks any access from external systems to the private network connection to the PLCs. In order to connect the RSLogix PLC programming tool to the PLCs, a cable must be deliberately connected between an MS WindowsXP computer and the private network switch.

OUTSTANDING ISSUES

The present PLC5 Device Support does not restore the PLC-IOC connection after a power interruption to the PLCs. This will be resolved by placing the PLCs on a UPS circuit so that brief interruptions do not affect the connection. In the near future a reconnect function will be implemented in the driver.

LESSONS LEARNED

The project was to take 6 months based on experience with typical new installations. It actually took about one man-year of work spread over 26 months.

The *EPICS Learning Curve* contributed to the long development time, as the author was new to the environment.

Much of the time was taken in building the databases and screens, and then fine tuning them to get proper operation. There was an extensive amount of infrastructure required, including network configuration, power distribution and IOC target platform research and configuration. New software tools included scripts to bring the project within the existing project control mechanisms, principally deployment of production files to the target system.

Unforeseen was the effort going into upgrading the PLC driver.

A final influence was due to resource conflicts with other high priority projects and the requirement to schedule work around the on-going isotope production schedule.

ACKNOWLEDGEMENTS

The author would like to acknowledge the following for their help in completing this project.

- B.Sidhu and J. Lofvendahl, TRIUMF Applied Technology Group
- TRIUMF ISAC Controls Group
- TRIUMF Nuclear Medicine Group
- J. Sinclair, Spallation Neutron Source

REFERENCES

[1] D. J. Dale, T. Ewert, D. Harrison, J. Lam, and R. Keitel, "The TR13 Control System for Automatic Isotope Production", ICALEPCS93, Berlin, 1993.
[2] http://www.advancedcyclotron.com/ [Online]
[3] Allen Bradley Corp. PLC-5/25, PLC-5/40L.
[4] Dynapro Systems Inc. Richmond, B.C. Canada
[5] J. Sinclair, Spallation Neutron Source, Oak Ridge, TN.
[6] R. Keitel, "Generating EPICS IOC Databases from a Relational Database - A Different Approach" ICALEPCS01, San Jose, 2001.
[7] http://www.phase3.com/ [Online]
[8] R. Keitel and R. Nussbaumer, "Automated Checking of Interlocks in the ISAC Control System", ICALEPCS01, San Jose, 2001.

SHAPE CONTROLLER UPGRADES FOR THE JET ITER-LIKE WALL

A. Neto, D. Alves, I. S. Carvalho,
Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear - Laboratório Associado,
Instituto Superior Técnico, Universidade Técnica de Lisboa,
1049-001 Lisboa, Portugal
P. J. Lomas, R. Felton, P. McCullen, V. Riccardo, F. G. Rimini, A. Stephen, K-D. Zastrow,
EURATOM-CCFE Fusion Association, Culham Science Centre,
Abingdon OX14 3DB, United Kingdom
F. Maviglia, G. De Tommasi,
Associazione EURATOM-ENEA-CREATE, Via Claudio 21, 80125, Napoli, Italy
R. Vitelli, Università di Roma, Tor Vergata, Via del Politecnico 1-00133, Roma, Italy
JET EFDA Contributors*, Culham Science Centre, OX14 3DB, Abingdon, UK

Abstract

The upgrade of JET to a new all-metal wall will pose a set of new challenges regarding machine operation and protection. One of the key problems is that the present way of terminating a pulse, upon the detection of a problem, is limited to a predefined set of global responses, tailored to maximise the likelihood of a safe plasma landing. With the new wall, these might conflict with the requirement of avoiding localised heat fluxes in the wall components. As a consequence, the new system will be capable of dynamically adapting its response behaviour, according to the experimental conditions at the time of the stop request and during the termination itself. Also in the context of the new ITER-like wall, two further upgrades were designed to be implemented in the shape controller architecture. The first will allow safer operation of the machine and consists of a power-supply current limit avoidance scheme, which provides a trade-off between the desired plasma shape and the current distribution between the relevant actuators. The second is aimed at an optimised operation of the machine, enabling an earlier formation of a special magnetic configuration where the last plasma closed flux surface is not defined by a physical limiter. The upgraded shape controller system, besides providing the new functionality, is expected to continue to provide the first line of defence against erroneous plasma position and current requests. This paper presents the required architectural changes to the JET plasma shape controller system.

INTRODUCTION

The JET [1] plasma position and current controller (PPCC) [2] is a real-time system responsible for the control of the currents in poloidal fields (PF) circuits, the plasma current and the plasma shape [3]. Its two main components are the shape controller (SC) and the vertical stabilisation (VS) systems. PPCC receives its input data from the magnetics systems, poloidal field (PF) circuits, central interlock and safety system (CISS), pulse termination network (PTN), radio-frequency (RF) and lower-hybrid (LH) systems, central timing and triggering system (CTTS) and Level-1, the high-level plant configuration interface. The latter is used for the configuration of the system, both with parameters from the session leader (SL) user-interface and from the PPCC expert settings. Magnetic signals arrive from the ATM real-time network systems [4] and are used to calculate the plasma current, radial and vertical moments, perform the reconstruction of the plasma boundary [5] and to provide flux measurements. Some of these are used later as control variables in the form of plasma current, gaps and flux. The PTN connection enables the shape controller system to change its execution configuration, as part of a global response to an external event.

The main output of SC is the voltage reference to all the PF amplifiers. It can also trigger the stop of a JET pulse through PTN, or CISS, in the case of internal problems, control errors and violation of amplifier limits. The boundary reconstruction coefficients and the circuit currents are sent to other JET plant using the ATM real-time network.

Each circuit can be controlled either in absolute current, proportional (to the plasma current) current or against a geometrical parameter, defined as a vector, named gap. Examples of gaps are the radial outer gap (ROG), which defines the distance between the outer limiter and the plasma, and the radial inner gap (RIG). The amplifier can also be set in a blocked state, where minimum negative voltage is applied, and no induced current is allowed to flow in the circuit (due to the presence of a diode); or in the free-wheeling state where 0 V are applied across the circuit.

The SC system is based on time windows, so that each circuit can change its control mode during the experiment. Indeed, most of the amplifiers start the experiment either blocked or in absolute current control, as no plasma exists at that time. As soon as a sustained plasma breakdown is achieved, most circuits are either set in proportional current or gap control. The shape controller hardware is based on

* See the Appendix of F. Romanelli et al., Proceedings of the 23rd IAEA Fusion Energy Conference 2010, Daejeon, Korea

VME®, PowerPC® and custom built I/O and data acquisition cards. The system control cycle is 2 ms.

eXtreme Shape Controller

A module named eXtreme Shape Controller (XSC) [6] allows a full boundary control. The main idea behind the XSC algorithm is to control the full plasma boundary, while minimising the error over a large set of geometrical shape descriptors, so that the system is no longer limited to the accurate control of only a few gaps, further constrained by the circuit control mode selections. When the XSC is used, all the circuits in shape controller are set to proportional current control, with the appropriate current references provided by the XSC algorithm.

The XSC operational scenarios are implemented around a given plasma shape and equilibrium (in terms of plasma current, internal inductance and the ratio of the poloidal kinetic to magnetic pressure). Using linearised plasma models, the predicted PF currents for a given plasma shape and configuration are evaluated. If these are within power supply saturation limits, the XSC is finally designed by carefully tuning the weighting matrices related to the minimisation function [7] and the system stability is assessed in closed-loop simulations. As the algorithm is valid only around a given equilibrium, the plasma must be driven into the reference conditions using shape controller, before enabling XSC.

P1 Logic

Plasma current is induced by changing the flux across a central solenoid. At JET this circuit is named P1E and is responsible for generating the current in the ten stacked central solenoid coils. The current is produced by a flywheel generator, named PFGC, capable of delivering a maximum of 400 MW of power. As shown in Fig. 1, at the same time, a second circuit, named PFX is also capable of driving current, in different windings, of the six central pancakes. The main role of this circuit is to reduce the stray fields, increasing the inboard vertical field and allowing for a more *D* shaped plasma.

The PFGC is a single quadrant amplifier, producing single direction current and voltage. A special circuit with hardware switches, named *s1* and *s4* in Fig. 1, allows for the current to flow in opposite directions during the same plasma pulse. When required, this scheme is used to provide a pre-magnetisation of the iron core, by forcing current through the solenoid in the opposite direction (*s1* closed and *s4* opened) in respect to the one using during the experiment (*s1* open and *s4* closed). The result is a larger amount of flux available to induce plasma current and consequently a greater pulse length.

Since having currents in opposite directions in the PFGC and PFX circuits will lead to high repulsion forces, up until recently, the current in the PFX circuit was only allowed to flow when the current in the PFGC circuit was greater than

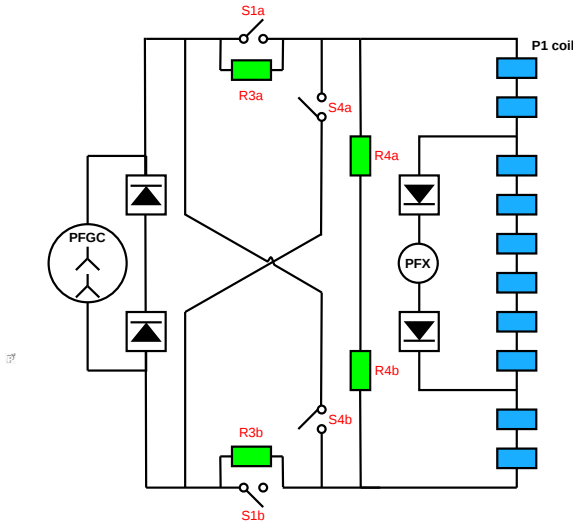


Figure 1: Simplified version of the JET ohmic circuit layout. By commutating the different switches, current from the flywheel generator will flow in opposite direction.

a positive value (with the convention that a positive PFGC current is the one to be used during the plasma pulse).

Justification for the Upgrades

The upgrade of JET to a new all-metal wall poses a set of new challenges regarding machine operation and protection. These will be handled by a major project named Protection of the ITER-like Wall (PIW), responsible for the coordination amongst all the infrastructure that provides the diagnostic data (e.g. infra-red cameras and pyrometers) and the systems required to take control actions (e.g. PPCC and additional heating systems). The major problem is that the current strategy to stop a plasma pulse might conflict with the requirement of avoiding localised heat fluxes in the inner wall components. The greatest limitation in PPCC was that the stop strategies are global during the pulse and could not be adapted for the pulse phase and plant conditions.

A second improvement, aimed at an optimised operation of the machine, was designed in order to enable a limited amount of PFX current to flow before a positive value of P1E current is achieved, without putting the machine integrity and safety in jeopardy. The enlarged operational space will allow the earlier formation and exploitation of the X-point configuration.

Finally, a current limit avoidance (CLA) [8] system was designed in order to avoid PF current saturations when the XSC is used to control the plasma shape. It uses the redundancy of the PF coil system to automatically obtain almost the same plasma reference shape with a different combination of currents in the PF coils. In particular, in the presence of severe disturbances, it tries to avoid the current saturations by *relaxing* the plasma shape constraints. Its feedback scheme asymptotically guarantees an optimal trade-off between shape error and distance of the coil currents from their limits.

PIW STOPS

As illustrated in Fig. 3 the PPCC code was updated to react to ten different stop sources for each available time window. Examples of stop sources are localised hot spots in the main chamber, magnetohydrodynamic modes and unavailability of a crucial resource for the experiment (e.g. additional neutral beam power heating). A stop response is configured with a control mode and a control value for each of the 9 PF circuits. The only exceptions are the P1 circuit, for plasma current control, and the P4 circuit, for radial wall distance control, which are programmed using an hold-and-ramp waveform. These waveforms consist of four distinct segments where an hold value time is specified, together with a ramp-rate to the following value. Moreover, the values and times of an hold-and-ramp waveform can be synchronised to the plasma current waveform values, being dynamically adjusted to its values at the time of the stop.

A special type of stop, named jump to termination, enables the system to react to a stop request by fast forwarding the execution to the time window corresponding to the pre-programmed termination time, ignoring any time windows in between.

The request for a stop is driven by the real-time protection sequencer (RTPS) [9], connected to PPCC using the real-time network. RTPS is responsible for the central coordination of all the JET actuators and reacts to alarm requests from the vessel thermal map (VTM) [10] system.

An alternative sequence facility is also available in the PIW PPCC version. The idea is to switch to a different experimental program if a given resource is not available during the execution of the experiment, allowing to maximise the optimisation of resources and machine time.

It should be noticed that the old, pre-PIW, stops are still available to be used when required and can be programmed together with the PIW stops.

POET

In order to study the implications of enabling current on the PFX circuit, while there is still current from P1E flowing in the opposite direction, a project named PFX-on-early-task (POET) was started. The major problems addressed concerned the modelling of the maximum forces in the P1 stack, taking into account the machine mechanical structure, and the effects of external faults in the machine integrity if these were to happen while taking advantage of the new operational space.

At the same time a study regarding the implementation of a new control logic for PPCC was also performed, mostly based on the value of the current across the central pancakes, given by: $P1C = P1E + PFX$. As depicted in Fig. 2, it was decided to set three operational regions: one where no PFX is allowed to flow, so that $P1C = P1E$; a second where the value of PFX is limited; and finally a region where the PFX value is only limited by the amplifier. Even if the control system is not used as a protection system, it makes every effort to avoid triggering the interlock protection systems, and consequently a state machine with the previously described logic was implemented in PPCC. All the possible current combinations, simulation of possible faults, and the performance of the controller itself were asserted using a simulator of the controller, connected to linearised plasma models provided by the CREATE [11] tools. The results of these simulations have also allowed to select the first set of values for the limits, even if these are then expected to be tuned during the experimental campaigns.

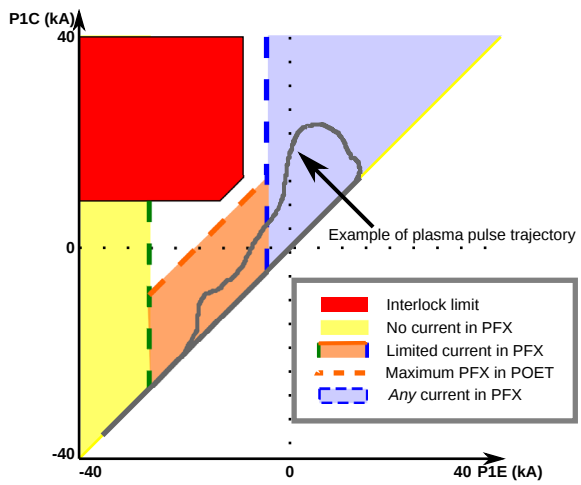


Figure 2: The new POET region will enable the use of PFX early in the plasma discharge. A possible plasma pulse trajectory is represented in grey. PPCC ensure that none of the limits are violated due to an erroneous request.

CURRENT LIMIT AVOIDANCE

The previously introduced current limit avoidance (CLA) was implemented as an additional feature of the eXtreme Shape Controller, being completely transparent to the existing SC code. As depicted in Fig. 4, the CLA modifies the PF current requests computed by the XSC before sending them to the SC, which is set in proportional current control mode with the only exception of the P1E circuit, as this is usually controlling the plasma current. Even when away from the saturation limits the CLA can be used to automatically equalise the best distribution of currents for a given shape, a process that is usually manually performed while developing the XSC controller settings for a given experiment.

Any changes performed by the CLA to the actual plasma shape are hidden from the XSC controller in order to inhibit its reaction. More details regarding the actual algorithm can be found in [8].

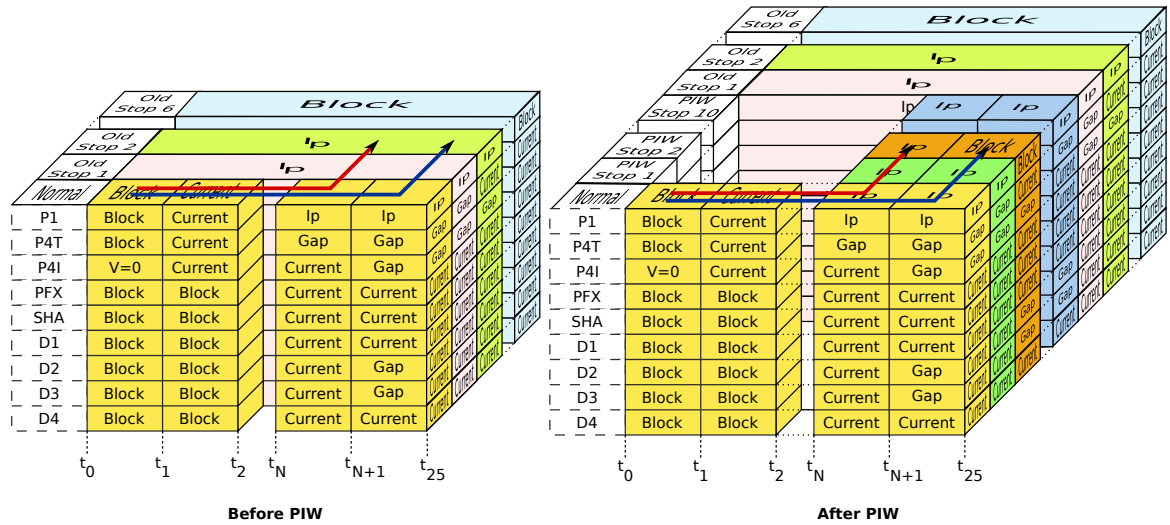


Figure 3: The previous version of shape controller had the same reaction to the same stop trigger, independently of the pulse phase and plant conditions. With the new all metal wall this could lead to highly localised hot spots and it was decided to allow for up to ten different stop responses for each experimental time window. The first column labels the 9 PF circuits controller by SC. In this example, for the same stop trigger (PIW stop 2), depending on the stop time, different control modes will be set.

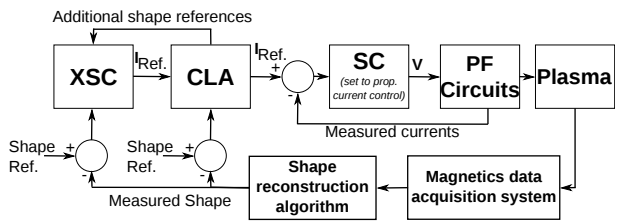


Figure 4: The XSC and CLA are implemented as additional modules of shape controller. XSC provides the current references to be controlled by SC which then sends voltage commands to the PF amplifiers. The additional shape references have the purpose of hiding from the XSC controller any changes to the plasma shape due to the CLA algorithm.

CONCLUSIONS

The JET shape controller was upgraded in order to cope with the operational requirement of the new ITER-like-wall. A new stopping architecture enables a greater control over the experiment and enhances the first line of defence for the machine protection. A new operational mode, named POET, enables an earlier exploitation of the X-point configuration, optimising experimental resources and time. Finally, it is expected that with the introduction of a current limit avoidance module it will be possible to demonstrate a safer operation of the machine, while using XSC, even in the presence of disturbances and when working near the power supply limits.

REFERENCES

[1] JET EFDA Contributors, "Special Issue on Joint European Torus (JET)", Fusion Science Technology, Vol. 53, No. 4, pp. 861-1227, 2008

[2] A. Cenedese et al., "Plasma position and current control management at JET", Proceedings of 42nd IEEE Conference on Decision and Control, pp. 4628-4633, Vol.5, 2003

[3] F. Sartori et al., "The Joint European Torus, Plasma Position and Shape Control in the World's Largest Tokamak", IEEE Control Systems Magazine, Vol. 26, No. 2, pp. 64-78, 2006

[4] R. Felton et al., "Real-time plasma control at JET using an ATM network", Proceedings of the 11th IEEE NPSS Real Time Conference, pp. 175-181, 1999

[5] L. Zabeo et al., "A new approach to the solution of the vacuum magnetic problem in fusion machines", Fusion Engineering and Design, Vol. 82, pp. 1081-1088, 2007.

[6] G. Ambrosino et al., "Design and Implementation of an Output Regulation Controller for the JET Tokamak". IEEE Transactions on Control Systems Technology, Vol. 16, pp. 1101-1111, 2008

[7] M. Ariola et al., "Plasma shape control for the JET tokamak: an optimal output regulation approach", IEEE Control Systems Magazine, Vol. 25, No. 5, pp. 65-75, 2005

[8] G. De Tommasi et al., "Nonlinear dynamic allocator for optimal input/output performance trade-off: application to the JET Tokamak shape controller", Automatica, vol. 47, no. 5, pp. 981-987, May 2011

[9] A. Stephen et al., "Centralised Coordinated Control To Protect The JET ITER-like Wall", this conference

[10] D. Alves et al., "The Software and Hardware Architectural Design of the Vessel Thermal Map Real-Time System in JET", this conference

[11] R. Albanese et al., "The linearized CREATE-L plasma response model for the control of current, position and shape in tokamaks", Nuclear Fusion, Vol. 38, No. 5, pp. 723, 1998

UPGRADE OF THE CLS ACCELERATOR CONTROL AND INSTRUMENTATION SYSTEMS*

E. Matias #, L. Baribeau, S. Hu, C.G. Payne, H. Zhang
Canadian Light Source, University of Saskatchewan, Saskatoon, Canada

Abstract

The Canadian Light Source is undertaking a major upgrade to its accelerator system in preparation for the eventual migration to top-up and to meet the increasing demanding needs of its synchrotron user community. These upgrades on the Linac include the development of software for new modulators, RF sections, power supplies and current monitors. On the booster ring the upgrades include the development of new improved BPM instrumentation and improved diagnostics on the extracted beam. For the storage ring these upgrades include fast orbit correct, instrumentation for use by the safety systems and a new transverse feedback system.

CLS ACCELERATOR

The CLS linear accelerator was originally constructed in 1964, with major upgrades and enhancements in the 1980s and again in 1999 when the machine was repurposed as an injector for the new synchrotron. The booster ring, storage ring and first round of synchrotron beamlines were constructed and commissioned between 1999 and 2003. The 1999 upgrade included replacing all of pre-existing CAMAC equipment with VME and the adoption of EPICS. The only legacy control equipment after 1999 was hard-wired electronics and Modicon Micro84 PLC.

In order to effectively offer top-up operation a series of upgrades focused on machine reliability, availability and reproducibility are currently under way.

Currently under construction at the CLS is a second linear accelerator designed to demonstrate the feasibility of producing Molybdenum^{99} using a particle accelerator. This accelerator is a standalone machine not integrated into the main accelerator control system.

COMMON ELEMENTS

Architecture

As shown in Figure 1, CLS utilises an EPICS based control system using EPICS Channel Access as the primary enterprise data-bus.

EPICS is utilised at the lower layer. High level accelerator control screens have been developed using EDM. The Matlab - Accelerator Tool Box is used for high-level on-line physics applications.

*Research described in this paper was performed at the Canadian Light Source, which is supported by the National Sciences and Engineering Research Council of Canada, the National Research Council of Canada, the Canadian Institute for Health Research, the Province of Saskatchewan, Western Economic Diversification Canada, and the University of Saskatchewan.
elder.matias@lightsource.ca

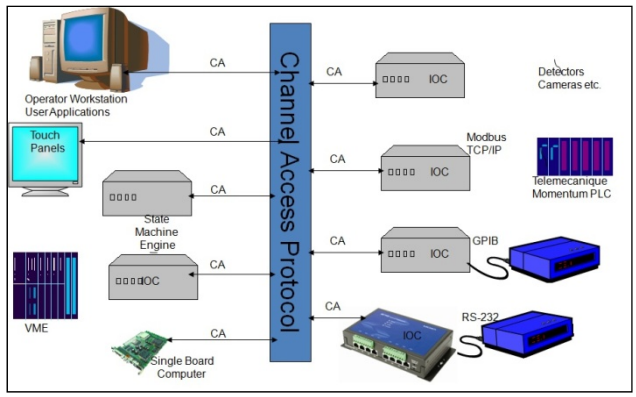


Figure 1: Architecture.

We find the use of the Moxa MIPS based controllers (Model UC-7408-LX and DA-662-16-LX) compelling [3]. Over the past four years CLS has deployed 120 Moxa computers in the facility and has yet to experience any significant hardware failure. The Moxas are used as full fledged EPICS IOCs primarily interfacing Ethernet, USB and Serial devices into the control system.

Increasing we are seeing a series of network aware Ethernet based devices being integrated into the control system.

Modicon Momentum PLCs continue to be used primarily for machine protection. These are interfaced into the CLS control system through Modbus over TCP/IP. Siemens S7/300 and S7/400 are used in the cryogenics plant, Booster Ring RF and Storage Ring RF Systems.

VME64x is primarily used for motion control and data acquisition applications. The Prodex MaxV is used with a combination of drivers [1]. Data acquisition is primarily accomplished with the use of SIS3820 scalar boards.

Some preliminary development work is occurring on the use of microTCA equipment, primarily focused on beamline data acquisition.

Alarm Handler

CLS has completed the transition from a locally developed legacy alarm handler to using the SNS BEAST[2] alarm handler. BEAST is built on top of the Control System Studio Toolkit [3]. The CLS deployment uses mySQL as the relational database for alarm storage.

Working closely with accelerator operations staff a strategy was adopted to limit alarms to those process variables that require action to be taken by operational staff.

CNSC regulatory requirements dictated that a proper human factors engineering review be performed using

relevant sections of NREG-0700. Based on careful review we were able to demonstrate compliance [4].

Currently under development is a second deployment of BEST targeting mechanical maintenance staff. These alarms would be restricted to process variable alarms that would indicate the need for preventative maintenance to occur.

The alarm handler is supplemented with a series of auto-diallers that are triggered based on certain critical process variables.

Electronic Logbook

After many years of relying on paper based log-books, CLS in the past year has started to introduce an electronic logbook based on the TRIUMF electronic log book system.

The electronic logbook system is currently operated in parallel and redundantly with the paper based logbook.

Save and Restore

CLS utilises a locally developed Save-Restore application. Originally driven by the need to support diskless IOCs, the CLS Save and Restore is a server based application that also permits accelerator operations staff to take and restore machine snap-shots of the accelerator configuration.

Radiation Monitoring

CLS has used the Canberra ADM606M radiation monitors for nearly eight years. The local indicators (horns and lights) were found to be inadequate given background noise in close proximity to the monitors. As a result of a human factors assessment of the monitors it was determined that the following two upgrades were required: (1) local annunciation needed to be augmented with the use of additional strobes and horns and (2) the control room interface needed to be upgraded based on human factors engineering standards [5].

An improved local annunciation system was developed and deployed. Data from the ADM monitors is obtained by EPICS running on Moxa UC7408 computers.

LINAC

The Linac is the oldest accelerator at CLS, originally constructed for nuclear physics experiment the Linac was repurposed in 1999 as an injector for the synchrotron facility. The machine has performed well over the past ten years, however given its age and the transition to top-up it is now necessary to upgrade the accelerator.

Linac ACIS System

CLS has used the Siemens S7/400 F PLC platform in the design of lockup systems for many years. However the Linac hall was the last remaining hall/tunnel to still make use of an older Modicon Micro84 based system. In October of 2009 CLS replaced the last remaining Modicon Micro84 system with a Siemens S7/400 system [6]. The system design is based on the requirements of IEC 61508.

The As-Low-As-Reasonably-Practical (ALARP) hazard and risk analysis technique was applied to the design of this system.

Linac Modulators

Work is ongoing in the replacement of end of life components on the linac, including linac sections, the gun, and modulators. New modulators were installed in April of 2011 with the installation of new sections expected in 2012.In the modulator upgrade, 6 old PFN type modulators were replaced with 6 solid state modulators from ScandiNova. Each modulator has 30 solid state IGBT's which discharge at 1170V and fire directly to a "split core" pulse transformer. The klystron pulse voltage is 250 KV, pulse current 250A, pulse width is 4 μs, and the pulse power is 62.5 MW.

Two MOXA DA662 computers are used for modulator monitor and control. Each computer communicates with 3 modulators using ScandiNova TCP/IP protocol, which is a variant of TCP/IP. StreamDevice is used for EPICS device support. The driver is based on an EPICS driver developed by PSI/Swiss Light Source.

A Modicon Momentum PLC is used to monitor vacuum and water status to provide external machine protection interlocks.

BOOSTER RING (BR1) ENHANCEMENT

Tbe booster is a relatively new machine that was installed in 2002.

Upgraded BPM Electronics

The CLS booster ring was commissioned using Bergoz BPM electronics. Once commissioned the need for BPMs diminished and they were not regularly used. As part of a plan to better characterise the injection system we decided to upgrade the booster to Libera Brilliance BPMs.

Eight Libera Brilliance BPMs have been installed in the booster ring to better understand the operation of the booster. The first measurement carried out with these Libera BPMs was to monitor the booster tunes. Figure 2 shows the booster tune measurement results at the beginning of the ramp, which were obtained by simple fast Fourier transform of the turn-by-turn beam position in each plane. To evaluate the performance of the booster during the whole ramp (600 ms), further attempts were made to monitor the tunes later in the ramp. Unfortunately, oscillations were not big enough to get a tune measurement at later times of the ramp. Therefore, the booster extraction kicker was used to excite the beam to introduce an oscillation at different stages of the ramp. In the past, the tunes at the extraction time were assumed to be the same as the tunes at injection, i.e., ν_x=0.806 and ν_y=0.77. The measurement results show a tune shift at the extraction time, ν_x=0.81 and ν_y=0.704. Recently, the booster response matrix was measured with stored beam in the booster at 150 MeV. Further efforts will be made to invert the response matrix with singular value decomposition and apply correction to improve the injection efficiency.

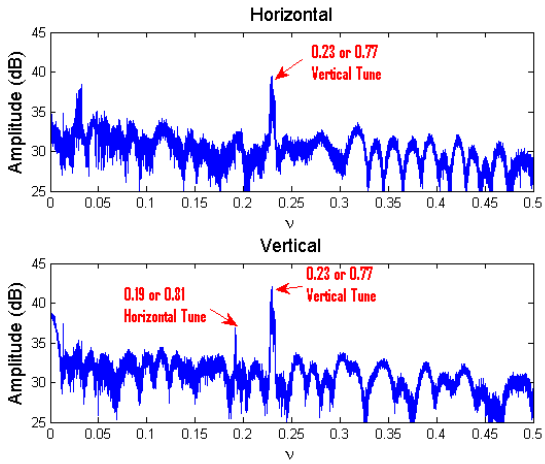


Figure 2: Booster Tune at the beginning of a ramp.

TRANSFER LINES

Power Supply Upgrade

Many of the original power supplies used in the CLS transfer lines were nearly 20 years old and dated back to the original EROS ring. CLS has recently completed a major replacement program to upgrade thelegacy power supplies with new power supplies from IE Power. The motivation for this upgrade was to reduce unplanned down-time and gain increased machine reproducibility, something critical as we move towards top-up operation.

Booster to Transfer Line BPM Monitor

CLS is currently testing a Libera Brilliance Single Pass BPM connected to strip-lines in the transfer line.

Current Transformers

CLS is currently installing additional Bergoz Fast Current Transformers (FCT) along the LTB transfer line connected to oscilloscopes. This augments a series of pre-existing Bergoz Integrated Current Transformers (ICT) that are connected to charge integrating ADC boards.

STORAGE RING (SR1) ENHANCEMENT

Transverse Feedback Systems

Due to the inherit stability of the storage ring, CLS was able to operate without the use of a transverse feedback system for many years. As the need for higher current operation became evident and user needs advanced, it became necessary to develop a transverse feedback system.

A transverse feedback system has been deployed to the CLS storage ring. It has three main functions:

1) Since high purity single bunch injection cannot be achieved at the CLS, any unwanted bunches have to be kicked out after injection. The transverse feedback system can work as a bunch cleaning system to establish an arbitrary fill pattern after injection. An enhanced feature, periodic bunch cleaning mode, has also been introduced to maintain high bunch purity, i.e., 10^{-5} at any time during a single bunch run.
2) The CLS transverse feedback system can be used to damp the transverse betatron oscillation associated with coupled-bunch instabilities. With the transverse feedback system, it is now possible to completely fill the storage ring, resulting in fewer electrons per bunch and consequently an increase in the beam lifetime. Figure 3 shows the comparison of the beam profile without/with the transverse feedback system.
3) Previously, the beam had to be excited by the injection kicker to determine the storage ring transverse tunes. With the transverse feedback system, it is now possible to excite only one bunch with limited amplitude for tune measurement. This feature has been extensively used for beam optics studies by CLS accelerator physicists.

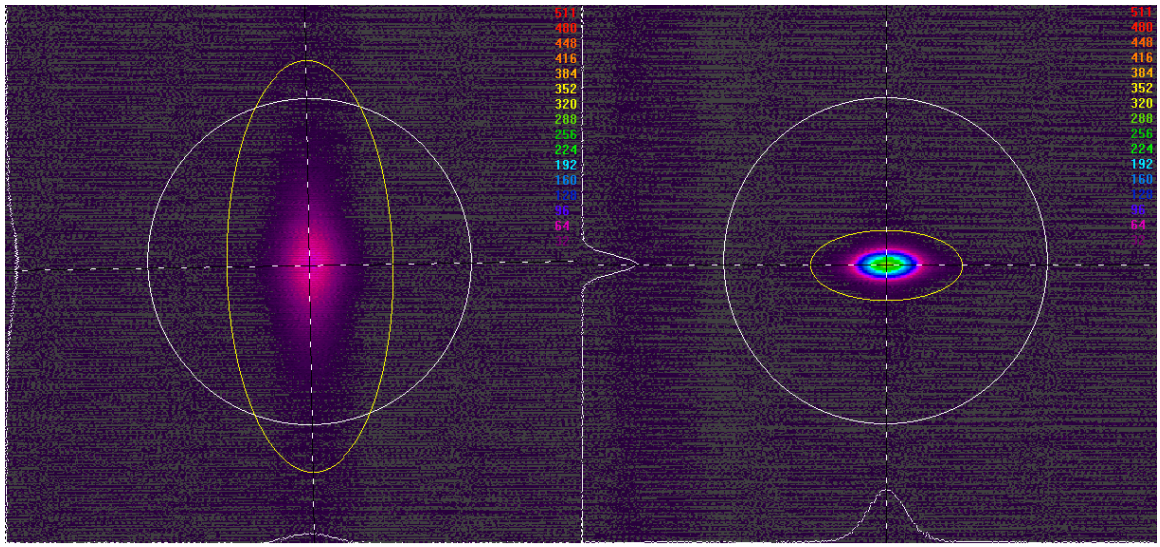


Figure 3: Transverse beam profile without/with TFB.

Orbit Correction

Due to the inherit stability of the machine CLS a fairly slow Matlab orbit correction scheme meet our initial needs. CLS is now in the process of commissioning a faster orbit correction system based on RTEMS.

CONCLUSION

Driven by the need to meet evolving requirements for increased machine reliability, availability and reproducibility we are undertaking a series of upgrades to the CLS accelerator control system.

REFERENCES

[1] D. Bertwistle, E. Matias, M. McKibben, "Experience with Motion Control Systems at The Canadian Light Source," ICALEPS'09, Kobe, Oct. 2009, WEP05, p. 501-503 (2009); http://www.JACoW.org.

[2] K. Kasemir, X. Chen. E. Danilova, "The Best Ever Alarm System Toolkit," ICALEPS'09 Kobe, Oct. 09, TUA01, p. 46-48 (2009); http://www.JACoW.org

[3] M. Clausen, J. Hatje, M. Moeller, H. Rickens, "Control System Studio Integrated Operating, Configuration and Development," ICALEPS'09 Kobe, Oct. 09, THC02, p. 667-669 (2009); http://www.JACoW.org

[4] L. Baribeau, "Analysis of Compliance to Human-Factors Interface Design Review Guidelines NREG-0700," CLS Technical Report No. 7.2.61.7 Rev. 1. (2011).

[5] L. Baribeau, "Active Area Radiation Monitoring System (AARMS) Design Manual," CLS Technical Report No. 7.9.52.3 Rev. 0. (2011).

[6] H. Zhang, E. Matias, G. Cubbon, C. Britton, R. Tanner, C. Finley, "CLS Linac Safety System Upgrade," PCaPAC'10 Saskatoon, Oct. 10, THPL010, p. 144-146 (2009); http://www.JACoW.org

[7] C. Payne, D. Chabot, "Fast Orbit Correction at The Canadian Light Source," PCaPAC'10 Saskatoon, Oct. 10, WACOMA02, p. 9-11 (2009); http://www.JACoW.org

ACSYS IN A BOX

C. Briegel, D. Finstrom, B. Hendricks, C. King, S. Lackey, R. Neswold, D. Nicklaus, J. Patrick, A. Petrov, R. Rechenmacher, C. Schumann, J. Smedinghoff

FNAL†, Batavia, IL 60510, U.S.A.

Abstract

The Accelerator Control System at Fermilab has evolved to enable this relatively large control system to be encapsulated into a "box" such as a laptop. The goal was to provide a platform isolated from the "online" control system. This platform can be used internally for making major upgrades and modifications without impacting operations. It also provides a standalone environment for research and development including a turnkey control system for collaborators. Over time, the code base running on Scientific Linux has enabled all the salient features of the Fermilab's control system to be captured in an off-the-shelf laptop. The anticipated additional benefits of packaging the system include improved maintenance, reliability, documentation, and future enhancements.

INTRODUCTION

The Fermilab control system [1] [2] has never been packaged for utilization by other institutions. When it was needed at remote sites, a special effort was made to provide an autonomous system with an umbilical cord to Fermilab to ease maintenance and operation. Until recently, there was never a need for a stand-alone implementation at an isolated site. New facilities [3] for accelerator R&D, collaboration with other institutions, remote experimental sites, and the desire to test new functionality in an isolated environment suggested the demand for packaging the Fermilab control system. The package with evolving enhancements is known as ACSys (Accelerator Control System) and has been demonstrated to run on a laptop.

REQUIREMENTS

This implementation is not intended to run on all operating systems and processors. The approach is to build for a rich environment while keeping the configurations manageable. Thus, a laptop running Scientific Linux is the foundation for a "single-box" solution. Additional "boxes" can be added as needed to interface hardware, increase the capability or isolate systems utilizing Scientific Linux or a vxWorks framework. Currently, Sybase is the supported database architecture.

PROCESS

The original goal was to provide a minimal control system implementing only the bare essentials. This reduced system was built on a Fermilab "standard-issue" laptop utilizing a scripting language, a device-only database, and a local front-end for data acquisition. While this version brought in much of the infrastructure, it left many desirable features out of the implementation.

The gradual extension of the goals evolved into a full-featured control system analogous to the existing Fermilab control system. The work to minimize the system proved to be more time-consuming in the long run and was not capable of maximizing the potential of the system. A full implementation enabled a platform for testing enhancements and provided an autonomous control system at remote locations.

ACSys is effectively the Fermilab control system with the ability to redirect the resources to coincide in a single instance. For protection of the existing control system, a unique IP port was used for communication. Also, distinct multicast addresses were used for various services such as state transitions and event notification.

COMPONENTS

ACSys is a tiered implementation (refer to Fig. 1.) It consists of a console tier for applications and displays; a central services tier for database, logging, save/restore, data consolidation and alarms; and a front-end tier for data access and control. These tiers are logical and can be associated with one or more physical pieces of hardware.

ACSys is a layered protocol (refer to Fig. 1.) Their is a peer-to-peer messaging protocol tunnelled on UDP/IP. This protocol is then used by several data acquisition protocols for reading and setting data, plotting data in real-time, and triggered data snapshots. Alarm handling involves another protocol providing unsolicited event and exception notifications.

Implementing layered protocols enables many implementations to coexist. For instance, data acquisition can be accomplished with multiple protocols in the same architecture. This enables the adiabatic evolution of enhancements without imposing change to stable or critical systems.

Console Tier

The console is a X-based server/client of displays and a client for data manipulation. These displays can be implemented in C++, Java, a drag-and-drop graphical display manager (Synoptic [4]), Web-based, or with a scripting language (ACL). The console optimizes data acquisition with a data pool manager for several console instances.

†Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

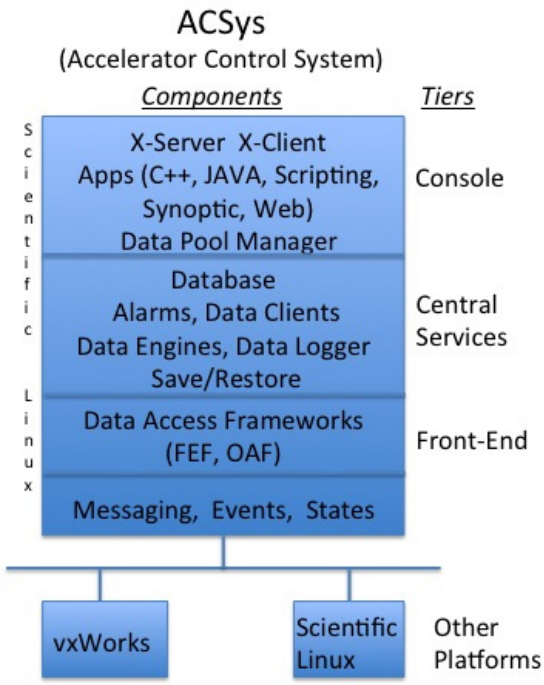


Figure 1: ACSys Configuration.

Central Services Tier

The database is a key component of the central tier. Sybase has been utilized for many years and the system capitalizes on several Sybase features. These features have been enumerated and ACSys provides an excellent development platform for an alternative database in the future.

The alarms server is a central repository for all announced alarms. These alarms are then distributed to consoles for individually configured displays. The monitoring of alarm limits is performed by a front-end. An unsolicited message is sent to the server when the condition becomes either bad or good.

The central tier provides a data engine to efficiently acquire and distribute data to clients such as save/restore. These engines also provide the equivalent data pool manager for Java applications.

Data loggers also use the engines to acquire a list of devices based on time or event. The data logger provides an archival circular repository. The duration of the repository is based on frequency and size of the data.

Front-end Tier

Prior to this effort, a new front-end framework (FEF) was being developed. This framework is designed for Scientific Linux. The framework is written in Erlang with C++ interfaces and is functional. A more mature framework called OAF (Open Access Frontend) is written in Java and has been used primarily for software devices or calculations.

The FEF [5] and OAF architectures complement other frameworks operating in vxWorks. These environments can be added to the configuration with additional platforms (outside the box).

Global Services

Other pieces of the infrastructure exist in ACSys. A peer-to-peer messaging system provides protocol payloads between cooperating connected tasks. Also, by setting a registered device, a global state transition is announced over a multicast frame. Normally, a hardware clock synchronizes data and its collection by asserting events in a serial protocol. A software solution can provide similar functionality generating periodic, sequenced and one-shot events. Global event notification occurs via a multicast frame for either hardware or software generated events.

The components listed above are not an exhaustive list of all available implementations, but are distinguished as core components. These components are readily available at this time to the implementer.

RESULTS

This effort has resulted in a fully capable stand-alone control system. ACSys provides a collaboration infrastructure consistent with the global control system at Fermilab. ACSys provides a testing environment for control system and accelerator system development. ACSys can also provide an isolated control system for remote operation.

A potential candidate for adopting the control system is an evolving experiment at Fermilab. In this case, the experiment has an on-site and remote detector. ACSys has positioned itself as a viable system to support this experiment. The control system offers common support and development with Fermilab personnel at a time of dwindling resources.

Current R&D efforts at Fermilab are using the Fermilab control system. A successful collaboration should ensure future instrumentation and analysis can be accomplished with Fermilab supported tools. ACSys makes remote R&D for collaborators a reality. This reality will aid in integration into the on-site control system providing a more maintainable system.

There are other benefits. Developing a generic system forces an understanding of dependencies that are sometimes taken for granted. These dependencies evolve since our target is not to commercialize the system, but to build a system for our users at Fermilab. Targeting other uses forces a reflection on implementation choices.

There are future benefits. The ACSys meetings and discussions are useful to discuss new directions and development. The direction of the group is providing input to new requirements of the control system. Hopefully, this direction will evolve into an active global development for the control system. The packaging of a system implies the documentation must be enhanced. This additional requirement is always a desirable bonus.

REFERENCES

[1] J. Patrick, "ACNET Control System Overview," Fermilab Beams-doc-1762-v1, http://beamdocs.fnal.gov/AD-public/DocDB/ShowDocument?docid=176.

[2] K. Cahill, L. Carmichael, D. Finstrom, B. Hendricks, S. Lackey, R. Neswold, J. Patrick, A. Petrov, C. Schumann, J. Smedinghoff, "Fermilab Control System," Fermilab Beams-doc-3260-v3, http://beamdocs.fnal.gov/AD-public/DocDB/ShowDocument?docid=326.

[3] S. Gysin, T. Bolshakov, C. Briegel, K. Cahill, L. Carmichael, D. Finstrom, B. Hendricks, C. King, W. Kissel, S. Lackey, W. Marsh, R. Neswold, D. Nicklaus, J. Patrick, A. Petrov, R. Rechenmacher, C. Schumann, J. Smedinghoff, D. Stenman, G. Vogel, A. Warner, "Project X Control System Requirements," Fermilab Beams-doc-2934-v22, http://beamdocs.fnal.gov/AD/DocDB/0029/002934/022/Ctrl_X-requirements.pdf.

[4] A. Petrov, Synoptic User Guide, http://beamdocs.fnal.gov/AD/DocDB/0035/003567/002/synoptic-1_2.pdf.

[5] D. Nicklaus, "Java Based Open Access Front Ends in the Fermilab Control's System," ICALEPCS (2003); Gyengjn, Korea.

REVOLUTION AT SOLEIL: REVIEW AND PROSPECT FOR MOTION CONTROL

D. Corruble, P. Betinelli-Deck, F. Blache, J. Coquet, N. Leclercq, R. Millet, A. Tournieux,
Synchrotron SOLEIL, Saint Aubin, France

Abstract

At any synchrotron facility, motors are numerous: they are the significant actuators of accelerators and the main actuators of beamlines. Since 2003, the Electronic Control and data Acquisition group at SOLEIL has defined a modular and reliable motion architecture integrating industrial products (Galil [1] controller, Midi-Ingenierie [2] and Phytron [3] power boards). Simultaneously, the software control group has developed a set of dedicated Tango [4] devices. At present, more than 1000 motors and 200 motion controller crates are in operation at SOLEIL.
Aware that motion control is important in improving performance, given that the positioning of optical systems and samples is a key element of any beamline, SOLEIL wants to upgrade its motion controller in order to maintain the facility at a high performance level and be able to respond to new requirements: better accuracy, complex trajectory and coupling multiaxis devices such as hexapods. This project is called REVOLUTION (REconsider Various contrOLlers for yoUr moTION).

CONTEXT AND HISTORY

Since 2003, the Electronic Control and data Acquisition group at SOLEIL has defined some standards (products and technologies) for motion control in order to provide high level support and maintenance while reducing costs.
One guideline is to prefer integration of industrial products rather than in-house development.

The challenge is to convince our internal customers that SOLEIL's motion control system can meet their specific needs with a ready solution.

The first implementation of this motion control system was done on LUCIA [5]; the first beamline of Soleil that was temporarily located in SLS [6] in late 2003. The first hardware and software prototypes were tested and debugged very early.

HARDWARE ARCHITECTURE AND PRODUCTS

To be modular and flexible, the hardware architecture of the motion control system is based on the following two principles. First the separation of control unit and power units, and then the use of standard signal exchanged between units, like a nerve impulse exchanged between the brain (control unit) and muscles (power unit). The motion units are packaged in a 19" rack that integrates industrial products. These racks specified by SOLEIL have the objectives of being easy to use, easy to maintain and cost-effective.

CONTROLBOX (the brain) integrates the Galil DMC-2182 8-axis industrial motion controller and provides easy to use connectors.

DRIVERBOX (muscles) integrates up to 8 Midi-Ingénierie boards dedicated to stepper motors.

VACUUMBOX (muscles) integrates Phytron power and control temperature boards, and is dedicated to stepper motors running in a vacuum.

SERVOBOX (muscles with a bit of mind) integrates a power board developed by SOLEIL and based on an Elmo [7] component: a numeric amplifier which manages a current loop. ServoBox is dedicated to servo motors (brushless and direct current). The prototype was validated in 2011

Today SOLEIL has standard hardware boxes able to control several kinds of motor: bipolar 4 phase stepper, brushless and DC. Theses boxes are also able to read several type of encoder: incremental (RS422/TTL signals), resolver, absolute SSI, and absolute analog.

Furthermore the flexible architecture could be connected to other technologies like 3 or 5 phase stepper motors, ceramic (piezo) actuators or sin/cos encoders via an interpolator.

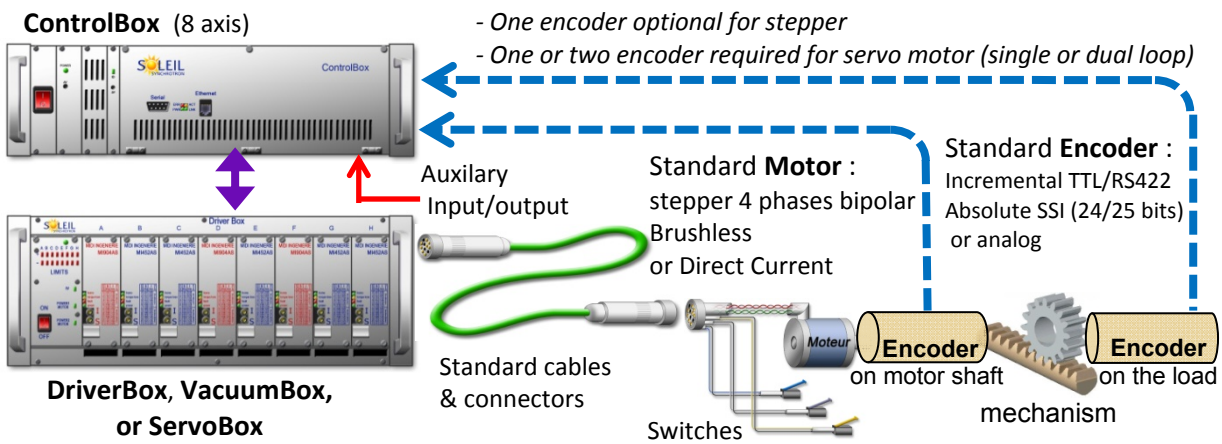


Figure 1 : Hardware architecture, standardized technologies and products.

To extend the range of supported technologies and add new features, a few years ago SOLEIL developed a flexible encoder tool in a SOLEIL proprietary format board [8]. The processing of encoder signals by a FPGA is used by example for encoder protocol transfer, encoder signals adder, complex trigger, etc.

SOFTWARE ARCHITECTURE: MULTI-LAYER PROCESSING

Motion control features are shared between embedded software in motion controllers and Tango [4] devices distributed on many servers.

Embedded Software in Motion Controllers

The firmware is provided by the controller manufacturer: Galil. It includes options for SSI encoders and for specific control loops dedicated to ceramic motors.

The microcode is developed by SOLEIL in a Galil proprietary language. Its first feature is to manage data exchanged with Tango device servers, including concentration of data and filtering all access by the device to the hardware functions. The second feature is to embed low-level motion processes as a discontinuous closed loop for stepper motors with backlash and progressive approach, static and dynamic error compensation, a multi-sensor homing process, and so on.

Theses functionalities can be completed by specific processes dedicated to specific applications especially for security of equipment. For example:

- Speed loop for phase control of RF cavities.
- Limitation of the range of 3-axis tripods according to the signals of inclinometers.
- Management of the ratio between powered and rest time of ceramic motor under vacuum.
- Limitation of the difference in position between 2 axes of a mirror bender.
- Collision avoidance according to security equations of multi-axis positioners.
- Etc.

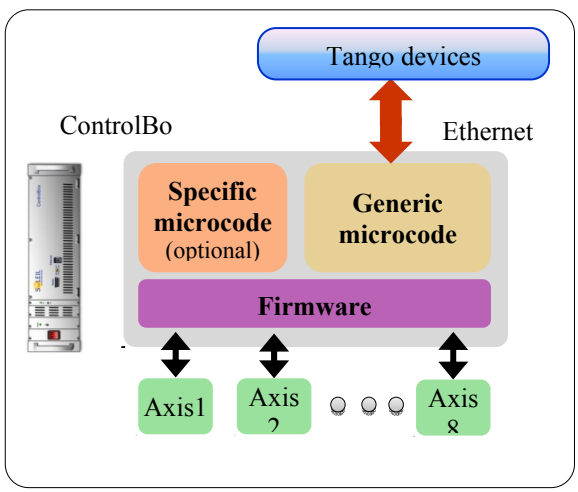


Figure 2: Embedded software architecture.

SOLEIL Motion Control Tango Devices Server

A set of motion control Tango device servers dedicated to our motion controllers has been developed in C++ by SOLEIL's Software Control and data Acquisition group. Binaries can run on Windows or Linux server.

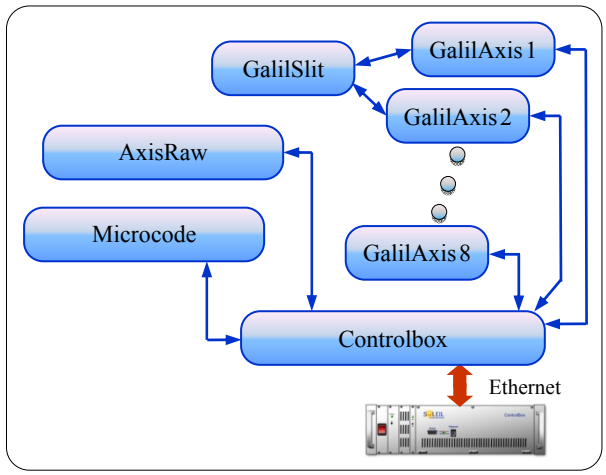


Figure 3: Tango devices servers architecture.

The lowest level and most essential device server is called ControlBox. It manages communication with hardware by TCP/IP protocol and provides data used for other devices. The GalilAxis device is a software object which represents a real motor. The device interface (properties, attributes and commands) is limited to simplify use. Multiple configurations and complex settings are hidden and not directly accessible to this software level. The GalilSlit device manages a pair of axes and supervises the master/slave relationship managed by hardware. It offers Gap and Offset attributes. The AxisRawDataReader device is a specialist device. It provides expert data from hardware, such as Tango attributes (read/only). Finally the MicrocodeDataViewer device provides access to variables of specific microcodes. It is used for supervision.

CURRENT RESULTS

Quantitative Results

Today the installed base of standard motion control equipment is wide and quite homogenous. It is broken down to around 15% for the machine and 85% for beamlines. As of May 2011, 1183 stepper motors controlled by 220 ControlBox and roughly as many DriverBox or VacuumBox units are operational at Soleil.

Furthermore 107 non-standard axes, like stepper 5 phases, ceramic and brushless motor (installed before ServoBox was in operation) are driven by 35 non-standard power racks, but they are controlled by ControlBox.

However, this is not the case for all motors. 37 mechanical devices with 244 axes are fully non-standard. They are mainly distributed in motorized magnetic insertion devices, diffractometers and hexapods. The reasons for this non-standard hardware, are firstly because

the control group did not have an effective solution when the equipment was chosen and also that manufacturers refuse to guarantee the performance of equipment if their own motion control system is not used.

Qualitative Results

Currently the initial objectives have been achieved and the motion systems work. Costs are controlled. There is no delay in providing working systems. The hardware components are reliable. Very little equipment is returned to the supplier. Software is efficient and reliable. Interventions relating to motion control during operation are very few. Performances are sufficient today for almost all applications.

Standardization has been largely successful. However, do not fall asleep and remember that future results get ready today: the revolution is coming…

MOTION CONTROL PROSPECTS: A REVOLUTION

Need to Upgrade: Birth of a Revolution

Firstly Galil released a new generation of motion controller (Accelera products) in 2009, so our standard controller belongs to the previous generation and has been produced for 8 years. It will be discontinued in a few years, but when?

Secondly, new motion control applications become more complex and demand more performance. Some examples:

- Hexapod platforms are more and more used in radiation facilities. They require control of complex multi-axis trajectories and calculation for kinematic equations.

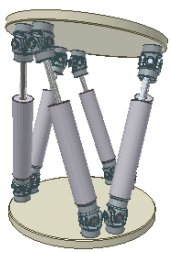


Figure 4: View of Stewart Hexapod platform.

- Nanopositioning: be able to position an axis to within few nanometers and keep this position by compensating for external disturbances, by using external sensor (e.g. interferometer). This requires fast and powerful calculations and openness to new technologies.
- Synchronization: be able to synchronize several axes and sensors to perform a continuous scan over complex trajectories. This requires fast signal processing calculations

So in the future, to preserve the use of standardized products, these products need to be improved. That means the next standard controller must be faster and more powerful in calculation and provide new features. SOLEIL must consider the next standard controller today.

A Technical Collaboration and a Founding Event

MAX IV [9] is the next Swedish radiation facility. The construction has just begun, the first beam is planned in 2015. MAX IV needs to define an up-to-date motion system architecture and select a controller quickly and efficiently.

SOLEIL and MAX IV have similar motion controller requirements: reliable, high performance and flexible, and they have similar guidelines too: minimal or no in-house development, standardization of hardware, modular solution, ready to use and complete solution

Consequently SOLEIL and MAX IV started a new technical collaboration to work together on this. This collaboration is called REVOLUTION, an acronym of REconsider Various contrOLlers for yoUr moTION.

In mid May 2011 SOLEIL and MAX IV organized a workshop "Exchange around motion control in radiation facilities". It was the founding event of the REVOLUTION project. 20 participants from 7 European facilities discussed an overview of motion systems in synchrotron facilities: status, main issues and plans

Figure 5: Map of radiation facilities participating in the motion control Workshop in May 2011.

The main results of the workshop are a written summary of thoughts [10], a creation of a dedicated mailing list "MOtion control Community in RAdiation Facilities" [11] and a principle of organizing a workshop every second year. The next motion control workshop will be hosted by the DIAMOND [12] facility, in the UK.

Some Guidelines of the REVOLUTION

The REVOLUTION's objective is to provide a complete ready-to-use solution for motion control. The main steps and deliveries of the project are clearly defined. The first and essential step is the selection of an industrial motion controller as a new standard. This controller will be integrated into a crate. After the design of the crate, an industrial company will be selected to manufacture it.

Another significant part relates to software developments: embedded software in the controller and Tango devices. The last step is about technical training for staff and writing technical documentation

The scoping of the project defines Rebox as the name of the next motion controller together with its crate. The capacity to replace any ControlBox by Rebox, and therefore backward compatibility of technical specifications (e.g. connectors) is a major principle. Finally the new features expected of the Rebox are set in the short, medium and long term (e.g. support of sin/cos encoder signals, extended auxiliary internal or remote I/O, etc.).

CONCLUSION AND PERSPERCTIVES

Although based on positive results in quality and quantity, the need for improvement of motion control systems is essential to maintain the SOLEIL facility at the highest level of performance. This need to obtain an efficient, complete and ready to use system is shared with the MAX IV facility under construction. REVOLUTION, the technical collaboration born from the common need will share costs and good ideas.

REVOLUTION is already underway. Starting from the motion control workshop, through the loan of equipment and skills with Swedish partners and continuing with a study of available controllers and detailed evaluation of some, the road is still long but major steps are defined to achieve the goal.

REFERENCES

[1] Galil supplier http://www.galilmc.com/
[2] Midi ingénierie supplier http://www.midi-ingenierie.com/
[3] Phytron supplier http://www.phytron.com/
[4] The TANGO framework http://www.tango-controls.org/
[5] LUCIA beamline http://www.synchrotron-soleil.fr/images/File/recherche/ligneslumiere/lucia/NIM_Lucia.pdf
[6] SLS radiation facility http://www.psi.ch/sls/
[7] Elmo supplier http://www.elmomc.com/
[8] A new set of electronic boards at Synchrotron SOLEIL. Paper WEMMU004 at ICALEPS 2011.
[9] MAX IV radiation facility http://www.maxlab.lu.se/maxlab/max4/index.html
[10] Workshop report http://www.synchrotron-soleil.fr/images/File/soleil/ToutesActualites/Workshops/2011/MotionControl/Report-Workshop-MoCRaF-2011.pdf
[11] Mocraf mailing list: mocraf@synchrotron-soleil.fr
[12] DIAMOND facility: http://www.diamond.ac.uk/

GDA AND EPICS: WORKING IN UNISON FOR SCIENCE DRIVEN DATA ACQUISITION AND CONTROL AT DIAMOND LIGHT SOURCE

E. P. Gibbons, M. T. Heron, N. P. Rees
Diamond Light Source, Oxfordshire, United Kingdom

Abstract

Diamond Light Source has recently received funding for an additional 10 photon beamlines, bringing the total to 32 beamlines and around 40 end-stations. These all use EPICS for the control of the underlying instrumentation associated with photon delivery, the experiment and most of the data acquisition hardware. For the scientific users Diamond has developed the Generic Data Acquisition (GDA) application framework to provide a consistent science interface across all beamlines. While each application is customised to the science of its beamline, all applications are built from the framework and predominantly interface to the underlying instrumentation through the EPICS abstraction. We will describe the complete system, illustrate how it can be configured for a specific beamline application, and how other synchrotrons are, and can, adapt these tools for their needs.

INTRODUCTION

Diamond Light Source has developed a beamline automation system that combines the low-level control system EPICS with the high-level data acquisition system GDA. This combination is used on the majority of beamlines at Diamond and has proven to be very successful at enabling scientists to maximize the quality and quantity of the wide range of science being performed. Use of the system results in a seamless experience for visiting scientists; from applying for beam time to having easily available data, be it raw, reduced, or partially analysed. The system design is such that changes in requirements can be accommodated in a timely and efficient manner. Both EPICS and GDA are open source projects allowing Diamond to benefit from development in both by third parties.

SYSTEM CONTEXT

The detailed description of the system begins with the generalised context diagram, shown in Fig. 1.

Scientist

Scientists interact with the system by reference to scientific properties of the experiment, such as the energy of the photons of the beamline. Scientists define experiments, run scripts and analyse data recorded. They do not have access to all low level details of the beamline.

Engineer

Engineers need access to all low level details of the beamline for commissioning and trouble shooting. The interaction is real-time in nature.

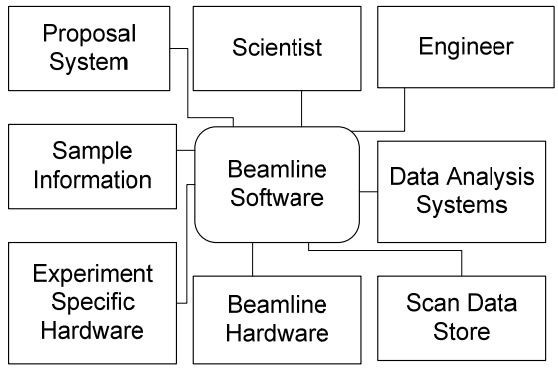


Figure 1: Context Diagram.

Data Analysis Systems

In order to aid scientists to make best use of time at Diamond great emphasis is given to providing on-site analysis of data. The aim is for this to be done automatically and the results fed back to the scientist as soon as possible.

Scan Data Store

The output of all scans is stored in a centrally available store. The store may also contain the results of any analysis performed on the data during the visit.

Beamline Hardware

This item encompasses all Diamond hardware on the beamline, as opposed to hardware brought by scientists for particular experiments. Hardware that is normally scanned or monitored during an experiment includes intelligent motor controllers, high speed 2d detectors and sample mounting robots.

It is important to note that for performance reasons some high speed detectors write directly to the Scan Data Store.

Experiment Specific Hardware

For some experiments scientists will make use of specialised equipment that is not normally on the beamline. Such equipment acts under control of the scanning mechanism of the beamline software. The beamline software has to be able to accommodate such additions in a timely and efficiently manner.

Sample Information

High level automation requires use of Sample Information systems containing details of samples and their locations in sample mounting robots. This information is presented to the scientist to aid automatic data collection, and linked to data to aid analysis.

Proposal System

This is Web based system used by potential users to request time on a beamline. The automation system uses this to check that a user is to be granted access to the beamline.

SOFTWARE DESCRIPTION

Figure 2 is a first level breakdown of the Beamline Software.

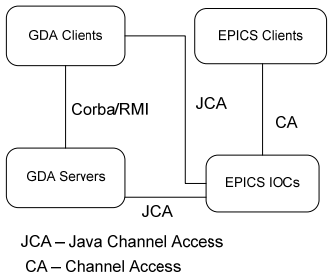


Figure 2: First level breakdown of the Beamline Software.

GDA Clients

Scientists interact with the beamline using the GDA Client program. This GUI is a Java application built using the Eclipse Rich Client Platform. It contains panels allowing the user to:

- Issue individual commands to perform actions. The command set is extremely diverse. A simple command would be to request the value of a variable, such as temperature readout. A complex command would be to execute a multi-dimensional scan of physical properties of the beamline resulting in measurements of multiple detectors and creation of a Scan Data file.
- Issue commands to execute a script file containing multiple commands. The script file can contain a simple list of commands or Jython programming language statements allowing the scientist to create complex sequences.
- Edit script files.
- Define scans using technique or even beamline specific GUI panels.
- View the results of scans and perform rudimentary analysis.
- Initiate science specific analysis of data.
- View results of automatic analysis of data.
- Perform technique specific actions using specific GUI elements, e.g. sample alignment.
- View the state of currently running measurements.

EPICS Clients

Engineers interact with the beamline using the EPICS clients, such as graphical user interfaces, which at Diamond are predominately EDM screens. EDM allows the beamline scientist or engineer to set and monitor values in the IOCs. EPICS archival and alarm handling systems are also used.

GDA Servers

The GDA Servers have several important features. Firstly the conversion between the engineering and scientific aspects of the beamline; this conversion is being represented by a set of Scannables and Detectors. Secondly a comprehensive and extensible scan mechanism for scanning such Scannables and Detectors and creating scan data files. Thirdly scriptable control of server objects via a Jython interpreter. The server can contain any additional Java objects one may want for a particular beamline; such as interfaces to hardware not supported by EPICS.

EPICS IOC

The EPICS IOC provide the majority of the hardware abstraction layer in the system, and specialised modules that are used by the GDA to provide important data acquisition tasks.

GDA AND EPICS IN UNISON

Given the very generic nature of GDA and EPICS there are many ways in which the two can be combined to create powerful, flexible and user friendly data acquisition systems. In the following sections we will describe two such systems in use at Diamond.

RECIPROCAL SPACE TRAJECTORY SCANNING

Beamline I16 at Diamond Light Source contains a six circle diffractometer which allows the orientation of a sample and detector relative to the incident x-ray beam to be set with few constraints. Such an arrangement permits measurements to be made over a wide range of reciprocal space.

GDA currently allows for three types of scans; a simple step scan, a constant velocity scan or a trajectory scan. In a simple step scan a measurement of a detector is made at different fixed motor, or pseudo-motor, positions. In a constant velocity scan the motors move at a constant velocity and the measurements are triggered by an additional time frame generator. In a trajectory scan the motors move along a series of points and the measurements are triggered to coincide with when each point is reached.

As well as performing scans of the six independent axes of the diffractometer, scientists need to scan in other space coordinates such as reciprocal space. A simple step scan through a number of points in reciprocal space is performed by the general purpose GDA scan command given in Fig. 3.

```
scan hkl [0 0 1] [0 0 1.1] [0 0 0.01] det1 0.1 peak2d
```

Figure 3: GDA command to perform a simple step scan through reciprocal space.

In the command the term ‘scan’ is the name of the command; ’hkl’ is the name of the object that represents a position in 3 dimensional reciprocal space; the 3 variables following ‘hkl’ represent the start, end and step size values of ‘hkl’ in the scan. The step size variable is followed by the name of the detector to be measured at each point in the scan and the exposure time per point. In this example the detector generates a 2d image; the final argument in this example is a pseudo-detector that analyses the reading from the 2d image and reports back the parameters of the analysis.

The system that relates hkl to diffractometer angle is called DiffCalc. Although built to work with the GDA scanning system DiffCalc can be used independently of GDA [10].

At each point in the scan the hkl object is asked to move to a new position represented by the tuple [h, k, l]. From these values the hkl object calculates the positions needed for the six axes and sends them as a single move command to the individual EPICS motor records in the appropriate IOC. Once all motors have moved the scan command then instructs the detector to read for the given exposure. After the exposure the value of hkl and the data from the detector are written to a scan file.

Trajectory scans are preferred to step scans as the former involve less time starting and stopping of motors. However trajectory scans need to be performed using low level motor controllers and involve hardware triggering of detectors and some form of data storage system. On the face of it, making a system for scientists to perform either scan from very similar generic commands would seem a daunting task; but given the design of the GDA scan mechanism this has been relatively straightforward.

To perform the same measurement as a trajectory scan the user simply replaces the term ‘scan’ in the command shown in Fig. 3 with ‘trajscan’.

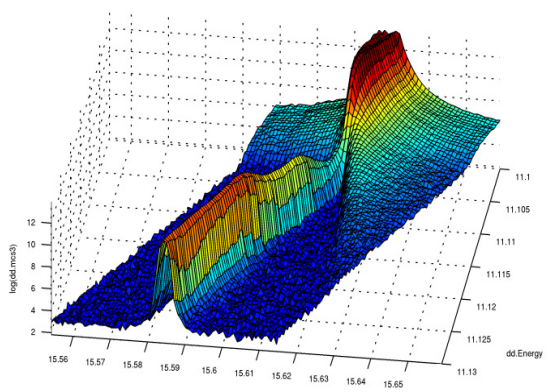


Figure 4: Detector readings from a trajectory scan of angle eta repeated for different incident photon energies.

In this case, rather than sending move commands to the six Epics motor records during the scan, the whole trajectory is calculated and sent to the related Epics TrajectoryScan [11] interface at the start. The Epics Trajectory Scan interface relays the trajectory to a program in the underlying intelligent motor controller. Once the scan is started the motor controller is responsible for sending the trigger pulses to the detector; GDA simply monitors the state of the detector as the scan proceeds. Whenever a new reading from the detector has been made the pseudo-detector peak2d fits a 2d Gaussian to the image and reports the size, position and sum over a region of interest. All of this data for each point is written to the scan file. Certain parts of the data are displayed on an XY plot in the GDA client.

Figure 4 displays actual results from using a trajectory scan of angle eta repeated for different photon energies. In this case the single channel detector was connected to a multichannel scaler with the motor controller’s at position signal connected to the channel advance source. GDA scanning is multi-dimensional which allows for the whole dataset to be taken by a single trajscan command.

Note that the trajscan command works with any scannables and detectors that implement particular Java interfaces. Also the list of detectors that can be read at each point is only limited by the particular hardware configuration. In addition the format of the data and how that is written is not limited. On I16 the Pilatus detector writes its image directly to the Scan Data Store and GDA simply records the filename in the scan data file along with the hkl values and the values from the analysis pseudo-detectors. On other detectors the whole image could be read by GDA and written directly into the scan data files. The scan data file format at Diamond is currently either a GDA specific ASCII format or, more commonly, Hierarchical Data Format (HDF5) compatible with the Nexus standard.

TIME RESOLVED 2D SCAN

Beamline I22 is a non-crystalline diffraction beamline at Diamond Light Source. A typical experiment on I22 involves recording a sequence of scattering patterns from a collection of detectors. The timing of the individual detector exposures is controlled by a time frame generator (TFG). For further complexity the whole sequence is repeated over a grid of positions of a sample.

Most of these detectors on I22 are connected to GDA directly. However a recent addition is a Pilatus detector that is interfaced to GDA using the EPICS AreaDetector system [12].

Once the TFG has been programmed via the GDA GUI the whole time resolved grid scan is performed by the single command shown in Fig 5.

```
scan px 0. 10. 1. py 0 10. 1. ncd
```

Figure 5: GDA command to perform a 2d scan of two motors, px and py, taking data from the detector system, ncd, at each point.

In the command in Fig. 5 px and py are ScannableMotors that control the x and y position of the sample and ncd is the detector system. Note that the ncd object starts the TFG for each value of px and py, and reads all the frames that are triggered by it.

The output of this command is a combination of 2 files. The first file, the so called master, is created by GDA, is compatible to the Nexus standard and is in HDF5 format. It contains the metadata for the scan, the positions of all scanned motors and the data from the directly connected detectors. The second file contains the Pilatus image data written by the HDF5 writer AreaDetector plugin; it too is in HDF5 format. The HDF5 writer plugin writes the data with a dimensionality that matches that of the data in the master file. By putting an external link to the second file in place of the Pilatus data in the first, Nexus compatible readers read the data in the second as if it were within the first.

HDF5 AREA DETECTOR PLUGIN

The use of the EPICS HDF5 AreaDetector plugin is proving very useful at Diamond, not only can much higher frame rates be achieved when writing to a single HDF5 file than the traditional approach of a single file per image, but the read rate is also improved.

CONCLUSION

The combination of GDA and EPICS has proven to be highly successful at providing flexible, powerful and user friendly data acquisition systems for beamlines at Diamond Light Source.

GDA is an open source project that is under continual development and welcomes collaborators. The www.opengda.org website provides downloads and documentation to aid evaluation by those considering its adoption.

ACKNOWLEDGEMENTS

GDA is primarily developed and supported by the Diamond Light Source Data Acquisition and Scientific Computing group. The EPICS system at Diamond is developed and supported by the Diamond Light Source Controls group. The following members of the two groups are acknowledged particularly: Mr R Walton and Dr M Pearson for the I16 trajectory scan; Dr T Richter for the I22 time resolved 2d scan; and Mr U Pedersen for the HDF5 AreaDetector plugin.

REFERENCES

[1] http://www.aps.anl.gov/epics/
[2] http://www.opengda.org/
[3] N.P. Rees, E.P. Gibbons, Diamond Controls Group, Diamond Data Acquisition Group. “The Diamond Beamline Controls and Data Acquisition Software Architecture”. SRI09. AIP Conf. Proc. (2010), Vol. 1234, pp. 736-739
[4] J.A. Thompson, I. Horswell, J. Marchal, U.K. Pedersen, S.R. Burge, J.D. Lipp and T.C. Nicholls “Controlling the EXCALIBUR Detector” ICALEPCS 2011.
[5] T.M. Cobb. “Integrating Gigabit Ethernet Cameras into EPICS at Diamond Light Source” ICALEPCS 2011.
[6] B.J. Nutter, T.M. Cobb, M.R. Pearson, N.P. Rees, F. Yuan. “Recent Developments in Synchronised Motion Control at Diamond Light Source” ICALEPCS 2011.
[7] R.J. Woolliscroft, C. Coles, M.R. Pearson. “Quick EXAFS Experiments Using a New GDA Based Eclipse RCP GUI with EPICS Hardware Control” ICALEPCS 2011.
[8] R. Mercado, I.J. Gillingham, J. Rowland, K.G. Wilkinson. “Integrating EtherCAT Based Remote IO into EPICS at Diamond” ICALEPCS 2011.
[9] M.G. Abbott, T.M. Cobb. “An EPICS IOC Builder” ICALEPCS 2011.
[10] R. Walton, “DiffCalc User Guide”; http://www.opengda.org/documentation/manuals/Diffcalc_User_Guide/trunk/contents.html
[11] M. Rivers, “EPICS trajectory Scan”; http://cars9.uchicago.edu/software/epics/trajectoryScan.html
[12] M. Rivers, “EPICS AreaDetector”; http://cars9.uchicago.edu/software/epics/areaDetector.html

TANGO COLLABORATION AND KERNEL STATUS

E. Taurel, ESRF, Grenoble, France on behalf of the Tango community
ALBA, DESY, ELETTRA, ESRF, FRM-II, MAX-LAB, SOLEIL

Abstract

This paper is divided in two parts. The first part summarizes the main changes done within the Tango[1] collaboration since the last Icalepcs conference. This will cover technical evolutions but also the new way our collaboration is managed. The second part will focus on the evolution of the so-called Tango event system (asynchronous communication between client and server). Since its beginning, within Tango, this type of communication is implemented using a CORBA notification service implementation called omniNotify. This system is being re-written using zeromq as transport layer. The reasons for the zeromq choice as well as a first feedback of the implementation will be given.

WHAT IS TANGO?

Tango is a control system tool kit developed by a community of several institutes. It is object oriented with the notion of devices (objects) for each piece of hardware or software to be controlled. Each device is an instance of a Tango class. Each Tango class is hardware or software specific. Tango classes are merged within an operating system process called a Device Server. Device configuration parameters and network addresses are kept in a database or in a file. Three types of communication between clients and servers are supported: synchronous, asynchronous and event driven.

KERNEL LIBRARIES

Since the Kobe conference, Tango has had 3 kernel library updates. The first release (Tango 7.1.1 in November 2009) was a minor changes and bug fixes release.

The second one was the release 7.2 in October 2010. The main change in this release is the thread safety of the client part of the Tango API. This means that you can share a C/C++ pointer to DeviceProxy instances between different threads. DeviceProxy is the name of the main Tango API client part class. Much faster algorithm when a device server process is shutdown was implemented. Another change is that an application (client) is now able to subscribe to the same event several times.

Then in March 2011, we had release 7.2.6 which was again a minor changes and bug fixes release.

PACKAGING, GUIS AND OTHER

Since several releases, Tango kernel libraries and basic tools are available for Linux via a source code distribution. It is based on classical GNU Autotools and allows a user to build and install the Tango control system with the standard configure / make / make install commands. For Windows, we provide a binary and ready to install distribution. Since several months, we have a binary distribution available for Linux as well. It is based on the Debian packaging system. The classical source distribution has been split into two source packages (for licensing issue related to our Java CORBA Object Request Broker) and 19 binary packages including documentation and debug packages. All these packages are available for Debian and Ubuntu linux flavours. For Ubuntu, a launchpad Personal Package Archive (PPA)[3] has been created making the Tango installation process a matter of a few clicks. The next Ubuntu release available end of October 2011 will natively incorporate these Tango binary packages in the Ubuntu Software Center.

Tango support three languages to write clients and servers. These languages are C++, Java and Python. We also have Graphical layers for these three languages. Since the very beginning of Tango, we have a Java layer called ATK (Application Tool Kit). This layer allows Java Swing application development with widgets (Java beans) interfaced to Tango objects (device, command or attribute). ATK is continuously developed by adding new widgets adapted to requests regarding graphical application development. We now have another Java GUI layer named Comète. It is developed by our Soleil colleagues. This layer opens the data source to something else than Tango objects. Using Comete, it is possible within the same application to get data coming from live Tango devices, but also from the Tango history database (Hdb) or from data files. See mini oral WEMAU012 for more informations on this subject. A C++ graphical layer named Qtango[3] and based on Qt[4] is also available. It is actively developed by Elettra and a online GUI development tool has been added recently. See poster WEPKS022. Finally, a Python layer named Taurus[5] is in active development at the Cells-Alba synchrotron. It is based on PyQt and is fully integrated in the Qt designer tool.

Our code generator named Pogo is since its major release 7 based on a Domain Specific Language (DSL) using the Xtext[6] and Xpand[7] technologies. It is now routinely used to create / update C++ Tango classes. Nevertheless, this tool in its release 7 does not support Python or Java Tango classes. You still have to use the previous Pogo release 6 in these cases.

Since the very beginning of 2011, the Tango security system is routinely used to protect the ESRF machine control system. This allows safer routine operations of the accelerator complex.

The Tango archiving service is actively developed and is now used in several institutes. See poster MOPKN016 for more information on this subject.

COLLABORATION MANAGEMENT

With the increasing number of collaborators using and/or developing Tango, it was becoming difficult to take decisions regarding its evolution. This could become a major problem in the near feature if nothing was done. Therefore, the rules governing our collaboration have been re-discussed and refined. We now have a new release of our Memorandum Of Understanding (MoU)[8]. Three types of collaborators are now defined. The first type are collaborators who do not sign this MoU. Not signing the MoU means that the institute (or individual) is not part of the Tango management board and does not have a right to vote on Tango issues. They are Tango users. The two other types of collaborators are the so-called contributors and committers. The committers contribute resources to the collaboration. The contributors can propose code modifications to the committers for Tango core issues and /or submits Tango device classes to the public repositories. Today (October 2011), we have 4 committers and 3 contributors.

All the strategic decisions about Tango development are now taken by an executive committee. This committee has one member for each institute who has signed the MoU (committers or contributors) plus a "collaboration coordinator". If there is no common agreement between all the committee members on a particular subject, decisions are made by voting. To take a decision, a 2/3 majority is required. Each committee member has at least a weight of 1. An extra vote is acquired for members representing committers institute.

The collaboration manager does not have voting right. His rule is to chair the executive committee meeting, to inform the collaboration of the strategic decisions made during the meetings and to discuss with Tango related project leaders matters like schedule and resources. A Tango executive committee meeting is organized at each classical Tango collaboration meeting. Examples of decision taken by this committee are:

- Tango is no more supported on Solaris platform
- The Source Code Management system used in public repositories related to Tango has to be SVN.
- The new Tango event system will be implemented using the zeromq software
- A list of 9 kernel improvements (extracted from a list of 27) has been selected as having the highest priority.

ON-GOING PROJECTS

On top of the classical evolution of the software already developed around Tango, we have several new projects in their development phase.

On the Java side, our colleagues from Soleil decided to take over the rewriting of the kernel part used when writing Java Tango class. Before this Soleil decision, we did not have the necessary resources within our collaboration to develop and maintain the Tango framework for our three languages at the same level. New features are implemented first in C++. Tango in Python is a layer above the C++ implementation and therefore benefits from the new C++ features with little effort. Java is an independent development and Tango kernel used in Java Tango classes were several major releases late compared to C++ and Python. The new development uses new features added in Java 1.5 like annotations which we hope will made the maintenance of this project less time consuming.

As already explained, the new major release of our code generator is not able to generate Java or Python code. Once the development concerning Java Tango classes will be finished, the code generator will be updated. For Python, more thinking has to be done about the best way to integrate this language in the code generator.

With the always increasing number of features incorporated within Tango, it was more and more difficult to achieve a good level of stability when a new release is introduced. To address this problem, we are now doing Continuous Integration using Jenkins[9]. The tool is configured in a way that as soon as we commit some changes in the repository of Tango C++ kernel code, it generates 20 different flavours of the libraries on 5 Operating Systems (mainly Linux and Windows). Then it compiles 10 Tango classes and finally run our test suite on the 5 operating systems.

Our test suite were composed of two different parts:

- Several test cases developed using a home made test system
- Other test cases using shell script and small C++ software with classical assertions.

A new project is now well on its way to merge these two blocks of test cases in a single one using CxxTest. CxxTest[10] is a xUnit like testing framework for C/C++. By adding new test cases, we will also try to increase our test coverage of the Tango kernel libraries to something close to 75%.

The work needed to implement the 9 kernel improvements selected by the Tango executive committee has also started. Here are some examples of these nine tasks:

- The new Tango event system (detailed below)
- The test suite improvements (shortly explained above)
- The need to have Tango device attribute with enumerated data type
- Implement structures as possible data type for Tango device attributes. This is a limited definition of structure: Only one level (no structure as data member of a structure) and all data members have to be simple Tango data type.

RE-THINKING THE EVENT SYSTEM

The Tango event system is based on the CORBA notification service. When an event is detected (or thrown by the user code), it is sent to the notification service. Then, it is the job of the notification service to forward

the event to all the processes which have subscribed to this event. We are using the CORBA notification service implementation called omniNotify. We now have some experience with this architecture and the following drawbacks have been detected:

- In the case of several clients (event consumers) interested by the same event, the notification service forwards the event to each client using unicast network transfer. This can be a bottleneck in case of a large number of consumers and in case of events carrying large amounts of data (eg images)
- The event data are transferred using CORBA Any objects. This means that in term of performance, it suffers from the unavoidable memory copy due to CORBA Any usage
- In case of event supplier sending events at a high rate with events carrying large amount of data and several subscribed consumers, the notifd has to buffer the event data. This could easily leads to a large memory consumption in the notification service process.
- The omniNotify implementation we have selected is an open source software but it is a "dead" project.

ZEROMQ

After miscellaneous studies, decision has been taken by the Tango executive committee to do a re-factoring of the Tango event system based on zeromq[11] software. The main point leading to this choice are:

- The high level of performance given by zeromq
- Its simplicity in term of infrastructure required (no additional process, no shared memory usage)
- Its availability in many different languages
- The support of a multicast protocol

But what is zeromq? This definition is taken from the zeromq guide available from their web site: *0MQ (ZeroMQ, OMQ, zmq) looks like an embeddable networking library but acts like a concurrency framework. It gives you sockets that carry whole messages across various transports like in-process, inter-process, TCP and multicast. You can connect sockets N-to-N with pattern like fanout, pub-sub, task distribution and request-reply. Its asynchronous I/O model gives you scalable multicore applications, built as asynchronous message-processing tasks. It has a score of language APIs and runs on most operating systems. 0MQ is from iMatix and is LGPL open source*.

During the pre-choice studies, the Data Distribution Service (DDS)[12] was also a candidate. Tango is definitively Open Source and this limits the number of DDS implementations available. Even if the level of performance given by DDS is also attractive (but less than 0MQ), it has been judged as less simple for the end-user. The tested implementation requires several additional processes to run on each host where it is used which make the system heavy. It also uses shared memory which could be damaged in case of process using DDS crashes.

THE TANGO USAGE OF ZEROMQ

0MQ provides a way to transport your data but it does not address the problem of data formatting for communication between computers built on different architecture. For synchronous and classical asynchronous communications (not event driven), Tango uses CORBA which has a well defined Common Data Representation (CDR). If the data you want to transport are defined in a CORBA IDL file, all the Object Request Broker (ORB) compilers will generate methods to encode or decode your data to/from this CDR definition. Therefore, the CORBA CDR is the data encoding selected for the new Tango event system while the transport is done using 0MQ.

We also need to define which data has to be transferred between the event supplier and the event consumer(s) to implement a full features event system. These data have been grouped in four parts:

- A string describing the event type: This string is built from the fully qualified Tango device name, the device attribute name and the event type (eg: tango://host:port/the/dev/name/attribute_name.change)
- A single byte encoding the event sender host endianess
- Some call informations allowing the Tango software layer to retrieve which object / method has to be called on the event consumer side. These informations are mainly the receiving object identifier (global information for a whole Tango system) and the method name. As explained previously, these data are encoded using the CORBA CDR
- The event data themselves. These data are already defined in the Tango IDL file for the synchronous communication. These event data are sent on the wire using the CORBA CDR.

Due to this splitting, we are able to use 0MQ multipart messages with one message part for each data group. A 0MQ multipart message is an entity which is fully transferred or not at all. Either you receive all message parts or none of them. Each part of the multipart message is itself managed like a simple 0MQ message.

The event propagation between the event supplier and one or several event consumer(s) is implemented using the 0MQ pub/sub pattern. The event supplier (the Tango device server process) is the publisher while event consumers (the Tango clients) are the subscribers. When you have several subscribers connected to one publisher, it is the 0MQ layer which takes the responsibility to propagate the data to all subscribers. By default this is done using TCP unicast communication. 0MQ also supports a multicast transport using OpenPGM[13] which is an implementation of the Pragmatic General Multicast (PGM) protocol. PGM is a reliable multicast transport protocol. Using multicast to transport Tango event seems a natural way. Nevertheless, it needs to solve the multicast address problem. Every host belonging to a multicast group will receive all the events sent to this group. For

instance, if you have only one multicast address, all the hosts with publishers/subscribers processes will see all the events flying in the system. If some of the events carry large amount of data, it will rapidly become a performance bottleneck. Ideally, one multicast group (address) should be assigned to each event but this will lead to a very high number of addresses. In a more realistic world, the number of multicast addresses available is limited and you have to find an algorithm to spread your events in these multicast group. It's not at all an easy task and depends a lot on which kind of data are generated by the controlled equipments. The decision has been taken to, by default still uses TCP unicast transport for the event propagation. Nevertheless, a Tango control system administrator will have the choice to use multicast transport when tuning the control system. A Tango control system property (configuration data) is defined to specify multicasting usage. This configuration parameter contains the multicast address, the port number and the list of event (device name/attribute name and event type) which should be propagated using this multicast group.

With 0MQ pub/sub pattern, subscriber(s) must set a subscription. This subscription is used by 0MQ as a message filter. Subscription are length-specified blobs. By default, a subscriber filters out all incoming messages. When the subscription is defined, the subscriber receives only messages beginning with the specified subscription buffer. We are using the first part (fully qualified event name) of the multipart message sent by the publisher as the subscription buffer. Thus, a client will receive only the events it is interested in even if on the device server side, the same publisher is used to publish several types of events for several devices.

IMPLEMENTATION

On the server side, the implementation uses two publisher sockets. The first one is used to propagate the heartbeat event. This event regularly sent allows client(s) to know that the device server process is alive, The second socket is used for the real events publishing and is used for all event types for all devices hosted by the device server process.

On the client side, the implementation uses 3 sockets. The first one is a subscriber socket connected to the publisher(s) sending heartbeat events. The second one is the subscriber connected to the real event publisher(s). During Tango event subscription, this socket is connected to the publisher supplying this event and a new subscription blob is associated to this socket. The third socket is a 0MQ Request/Reply socket pair using in-process communication. The 0MQ Request/Reply pattern covers the classical case of one requester asking a service to do something and to send a reply to the requester. 0MQ sockets are not really thread safe. You can use them within different threads only if a full memory fence has been executed before its usage in another thread. The Tango API is thread safe. Therefore, we have to cover cases where several threads required Tango events subscription. As explained above, this requires some actions on the event subscriber socket. Therefore, the Tango event subscription is done via a Request/Reply socket couple with in-process transport.

We have selected release 3 of 0MQ because it implements subscription forwarding. This means that the subscription requests are forwarded to the publisher and the associated filtering is done on the its side. This leads to less network bandwidth usage and less CPU consumption on the subscriber side (client side).

0MQ is written in C/C++ but it's API is C. Nevertheless, a C++ binding is provided and used in the C++ Tango implementation. (thus also covering the Tango Python case). On Java side, 0MQ also provides a binding based on the Java Native Interface (JNI).

Some very preliminary performance tests have been done. The result are summarized in table 1. This is the number of event/sec for events carrying 1 32 bits integer and 1024 integers (32 bits as well) forwarded to 1 and 10 subscribers. Tests have been done using unicast transport. The publisher runs on a Intel core 2 duo at 2.6 Ghz. The subscribers run on a Intel P4 at 2.3 Ghz. The network bandwidth is 100 Mbit/sec.

Table 1: New Event System Preliminary Tests

	1 Long (32 bits)	1 K Long (32 bits)
1 Subscriber	25500	2150
10 Subscribers	2700	270

CONCLUSION

From the first part of this paper, it is clear that Tango is still evolving. The community still wants to improve it and the problem is not a lack of ideas on how it could be improved but rather a lack of resources to improve it. Concerning Tango event re-factoring, it is still too early to draw conclusions on 0MQ usage in a long term. Nevertheless, we now have in labs a Tango event system based on 0MQ. It gives a significant improvement in term of performances and allow Tango to be more user friendly by removing the need of one extra process (notifd) .

REFERENCES

[1] http://www.tango-controls.org
[2] https://launchpad.net/~tango-controls/+archive/core
[3] http://www.elettra.trieste.it/~tango/docs/qtango/doc/html/index.html
[4] http://qt.nokia.com
[5] http://www.tango-controls.org/static/taurus/latest/doc/html/index.html
[6] http://www.eclipse.org/Xtext/
[7] http://wiki.eclipse.org/Xpand
[8] http://www.tango-controls.org/about
[9] http://jenkins-ci.org/
[10] http://cxxtest.tigris.org/
[11] http://www.zeromq.org/
[12] http://portals.omg.org/dds/
[13] http://code.google.com/p/openpgm/

BLED: A TOP-DOWN APPROACH TO ACCELERATOR CONTROL SYSTEM DESIGN

J. Bobnar, K. Žagar, COBIK, Solkan, Slovenia

Abstract

In many existing controls projects the central database/inventory was introduced late in the project, usually to support installation or maintenance activities. Thus construction of this database was done in a bottom-up fashion by reverse engineering the installation. However, there are several benefits if the central database is introduced early in the machine design, such as the ability to simulate the system as a whole without having all the IOCs in place, it can be used as an input to the installation/commissioning plan, or act as an enforcer of certain conventions and quality processes. Based on our experience with the control systems, we have designed a central database BLED, which is used for storage of machine configuration and parameters as well as control system configuration, inventory, and cabling. The first implementation of BLED supports EPICS, meaning it is capable of storage and generation of EPICS templates and substitution files as well as archive, alarm and other configurations. With a goal in mind to provide functionality of several existing central databases (IRMIS [1], SNS db, DBSF etc.) a lot of effort has been made to design the database in a way to handle extremely large set-ups, consisting of millions of control system points. Furthermore, BLED also stores the lattice data, thus providing additional information (e.g. survey data) required by different engineering groups. The lattice import/export tools among others support MAD and TraceWin tools formats which are widely used in the machine design community.

INTRODUCTION

With the complexity of new machines growing, the complexity and size of the control systems also grows rapidly. Big machine control systems are normally distributed across several hundred I/O controllers, and each of them might run several different applications. In addition, the controllers are interconnected with numerous cables and to various kinds of equipment.

In such a large system it is recommended that an inventory of all the equipment and control points is maintained in a well organized structure, so that information is conveniently available. A good equipment and data management system can help during the design of the machine as well as developers and engineers during the development cycles. On the other hand the physicists and operators can benefit significantly from such a system during commissioning and operations.

BOTTOM-UP VS. TOP-DOWN APPROACH

Until today, several different systems have been set up in various laboratories around the world. Most of these systems are specific to each individual machine. They have usually been introduced late in the project, thus being created in a bottom-up approach: gathering all the information in the system and then the database was populated by reverse engineering the installation. Furthermore, the majority of the systems covers only one specific domain, such as for example inventory or lattice, depending on who initiated the use of the database and what stage the project was in at that time.

Having this in mind we wanted to create a general system, which would allow one to create the machine in a top-down approach, starting already in the design phase of the project and later being used during day-to-day operations.

This approach brings several benefits. Besides having a good overview of what has already been completed, it also allows simulating the system as a whole even before having all the parts in place. In addition, it can also serve as a data exchange system between developers and different types of development and design tools.

The database can serve as an input to the installation and commissioning plan. Well designed structure and tools can use the information about the machine and output it in a way required during each project phase. For example, at the very beginning it can be a powerful tool to enforce the naming convention for the whole facility, while later it can provide relevant data for engineers etc.

During the operational phase of the machine, the database serves as an image of the complete system and allows for an easy system-wide modification and change tracking of the installation.

MAIN OBJECTIVES

The first version of BLED was initiated by the European Spallation Source (ESS) [2]. ESS, currently in the design update phase, uses several different lattice design tools. Therefore, it was very inconvenient to share the data among the designers, since some use TraceWin [3] while others use MAD [4] as their lattice design tool. Having proper import/export tools for the database, the exchange of data can become much easier, because all users have access to all parts of the lattice design.

Beside the aforementioned benefits of top-down approach and data exchange, BLED shall also be used during construction to generate the survey data for engineers. Using the information that is provided by the lattice design tools, most of the required data is already

available and only needs to be exported in the appropriate way.

During control system development and operation the database stores the data for all the control points in the system and it is possible to generate configuration files for control system, such as EPICS db files or BEAST and BEAUtY XML configurations [5]. In addition to pure control system configuration, other system description files could also be obtained from the database (e.g. XAL XML configuration [6]).

The database can also be a valuable tool to non-engineering departments (e.g. purchasing department), which could use it to track the list of components that remain to be purchased as well as to keep track of the purchases (i.e. delivery dates).

Another important aspect of the database is the naming convention, which is advised to be introduced early in the project. Selecting and enforcing consistent convention presents a high organizational risk, because all teams must conform to the convention. With introduction of BLED, it is the database and the tools that will take care of properly named items and verify every entry.

DATABASE STRUCTURE

Database has been designed as a Java object model equipped with Java Hibernate [7] annotations. Once the model is created, Hibernate takes care of the construction of the complete database (and later communication), thus making the model completely independent of the underlying database technology. Initially ESS will use MySQL, but if at any later time a decision is made to switch to a different database, only the database driver needs to be replaced and Hibernate will generate a new database.

The database is designed to provide functionality to store the history of the entities, which are expected to evolve during development and operation. Every change in any of these entities is recorded in order to be able to at any time revert to a previous version.

Since no database can satisfy the needs of every user, special custom tables have also been defined, which can be used to create virtual tables and store additional data, which may not belong to any of the existing tables.

Current version of the database consists of approximately 60 tables divided into 6 packages.

The Main Package

The main object of the model is the *Subsystem*, which represents every single part of the machine, either be it a logical system such as the *Target* or a real system such as an *RF Cavity* or a *Beam Position Monitor*. The subsystems are organized hierarchically, where each subsystem can have none or several child elements. Within each single subsystem all children are sequentially ordered, considering their physical position in the machine. This allows for organizing all the accelerator components according to their location and/or functionality (e.g. *spoke* subsystem can have several *spoke cavities* subsystems as its children).

The Devices Package

Subsystem represents the base class for all components of the machine. Several other classes (e.g. *Magnets*, *Power Supplies*, and *Cavities*) that describe physical components are derived from this base class. These components form the actual machine. In case of the accelerator the devices within each parent subsystem are ordered either into the lattice (if applicable) or bound to a specific device, which they are related to (e.g. *Power Supply* is related to a *Magnet*, but is not a part of the lattice). Beside the machine related parameters, the database also contains information about the precise location and orientation of the components, allowing for generation of survey data or for simulating the particle beam.

The hierarchy of the components is such that one can easily add another type of device and create appropriate tables for it.

The Control System Package

Control system package defines all controls software related items. On one hand it provides the list of all control points, associated with appropriate subsystems, and on the other hand it provides information about installed configurations for archiving and alarm systems.

The initial version of BLED is primarily focused on the EPICS framework. Therefore, it contains the collections of all process variables, EPICS applications and templates, substitution values etc. as well as provides links to the versioning system (Mercurial), where the source code of the related software (i.e. device support) is located.

The Infrastructure & Deployment Package

All *IOControllers*, *Racks*, *Servers,* and *Computers* are also parts of BLED. Together with the infrastructure parts (*Room* and *Building*) they present the deployment view of the machine. The deployment view describes on which location and on which equipment the particular control system software is deployed.

The Wiring Package

Wiring is another important part of the installation. The database provides facilities to record each cable that connects any two ports in the machine. It is possible to define the type of the cable (including physical characteristics such as diameter) as well as the type of connector on all free ends of the cable. Not only linear (2-end cables) are supported, but also storage of Y or even more complicated cables is allowed.

The Parameters Package

Every machine or individual parts of it have certain characteristics, which are not physical descriptions of particular equipment, but represent higher-level information such as energy or minimum spot size on the target. BLED provides necessary tables to store such parameters and associate them with subsystems, responsible personnel, references etc. [8].

DATABASE INTERFACE TOOLS

In order to effectively manage and make the most use of the database, certain graphical tools will be developed. These tools will verify that each new entry in the database corresponds with the rules set forth by the designers. Using only such designated tools it will be assured that the database integrity is not broken during updates.

Besides the management tools, the system will also provide a set of tools necessary to generate particular configuration used either in the control system or any other part of the machine operation. Such tools represent the main interface to top-down development approach.

All management and configuration tools will be based on the web technologies (*Google Web Toolkit*). This will eliminate problems with running the software on different platforms as well as make it available to all registered users. Additional tools or updates to existing ones will not require users to do individual updates, since all the deployment will reside on the server.

Naming Convention Tools

Naming convention is an important issue, which has to be introduced early in the design of the machine in order to enforce it on every possible step. The naming convention tool will help the developers construct names for their control points so they will be in coherence with the facility-wide convention. The tool will allow the user to specify which subsystem a particular control point belongs to and suggest the appropriate name for it, based on the hierarchy of the subsystem.

Lattice Generators

During the design development of the machine several lattice design tools might be used. The interoperability of those tools is limited, because each of them works on special set of configuration files that are valid only for each particular design tool. In order to ease the process several parsers and data exporters will be created, which will parse these configuration files and import data into the database. Other users may then export the data into format understood by their preferred design tool.

Control System Configuration Management

ESS is estimated to have an order of one million control points. Construction and maintenance of such a large system can introduce significant problems if it is not managed and overseen properly. BLED tools will minimize these problems by verifying the completed work and help generating the missing parts of the system. Once the control points are defined, BLED will take care of generating configuration files for services (archive, alarm etc.), eliminating the risk of mistyping names or skipping a particular part of the system. Also, the effort required to do the task will be minimized.

At some time certain control points may change, effecting numerous configurations across the whole system. By having a well organized database, such cases can be easily spotted and all the necessary configurations would be regenerated.

Equipment Inventory

As mentioned before, BLED will also serve as the equipment inventory database. Therefore, tools will be created to allow editing as well as browsing and searching through the list of available equipment. It will display the details of each selected piece of equipment including its connections to other pieces of equipment and also allow tracking the location of the component.

Machine Parameters Management

Parameters management tools will assure that the parameters list is alive and that people responsible for each section will have access to the lists and be able to inspect and modify them [8]. Several criteria for searching the parameters will be established, such as for example subsystem or team criteria. In addition, facilities to export the data in various formats (Excel, PDF) will be provided.

CONCLUSION

Maintaining accurate information on several ten thousands of devices and a million control points can be a challenging work and cannot be done without proper tools. Therefore, a good management system needs to be installed and supported by extensive set of graphical and utility tools, which can save a lot of time and reduce risks.

ACKNOWLEDGEMENTS

We would like to express our gratitude to all the people involved on the BLED project, especially controls group at ESS, which was the main initiator of the project and funded the first developments.

REFERENCES

[1] Integrated Relational Model of Installed Systems – IRMIS (http://irmis.sourceforge.net/)
[2] European Spallation Source – ESS (http://ess-scandinavia.eu/)
[3] Ions and electron beams trace calculation tool – TraceWin (http://irfu.cea.fr/Sacm/logiciels/index3.php)
[4] Accelerator design and simulation tool- MAD (http://cern.ch/mad/)
[5] Control System Studio – CSS (http://ics-web.sns.ornl.gov/css/devel.html)
[6] XML Accelerator Model – XAL (http://sourceforge.net/projects/xaldev/)
[7] Object-relational mapping library – Hibernate (http://www.hibernate.org/)
[8] K. Rathsman et al, ESS Parameter List Database and Web Interface Tools, IPAC 2011 Proceedings

WEB-BASED EXECUTION OF GRAPHICAL WORKFLOWS: A MODULAR PLATFORM FOR MULTIFUNTIONAL SCIENTIFIC PROCESS AUTOMATION

E. De Ley, D. Jacobs, iSencia Belgium, Ghent, Belgium
M.Ounsy, Synchrotron Soleil, France

Abstract

The Passerelle process automation suite offers a fundamentally modular solution platform, based on a layered integration of several best-of-breed technologies. It has been successfully applied by Synchrotron Soleil as the sequencer for data acquisition and control processes on its beamlines, integrated with TANGO as a control bus and GlobalScreen™ as the SCADA package. Since last year it is being used as the graphical workflow component for the development of an eclipse-based Data Analysis Work Bench, at ESRF.

The top layer of Passerelle exposes an actor-based development paradigm, based on the Ptolemy framework (UC Berkeley). Actors provide explicit reusability and strong decoupling, combined with an inherently concurrent execution model. Actor libraries exist for TANGO integration, web-services, database operations, flow control, rules-based analysis, mathematical calculations, launching external scripts etc.

Passerelle's internal architecture is based on OSGi, the major Java framework for modular service-based applications. A large set of modules exist that can be recombined as desired to obtain different features and deployment models. Besides desktop versions of the Passerelle workflow workbench, there is also the Passerelle Manager. It is a secured web application including a graphical editor, for centralized design, execution, management and monitoring of process flows, integrating standard Java Enterprise services with OSGi.

We will present the internal technical architecture, some interesting application cases and the lessons learnt.

INTRODUCING PASSERELLE

Passerelle[1] is a component-based solution assembly suite, developed and distributed by iSencia Belgium since 2002. It is based on an integration and on extensions of Ptolemy II[2] (Univ. Berkeley), OSGi/Equinox & other OS libraries. Passerelle comes with an execution engine, enhanced actor development API, reusable actor libraries and several graphical model editors.

Passerelle is already used for several years in "production-mode" at different sites. Two important reference sites are :

- Belgacom, the main Belgian telecommunication provider, where it is used to automate diagnosis and repair processes for the customer help-desk operators, and for field technicians.
- Synchrotron Soleil in France, where Passerelle is used as sequencing engine for beamline experiments.

Since last year, the core Passerelle modules are provided in open source, together with an eclipse-based graphical editor, at the eclipse labs hosted by Google.

ACTOR-BASED DEVELOPMENT

Passerelle inherits many core concepts from Ptolemy II, the main ones being :

- actor-based development
- using graphical hierarchical models for defining executable solution assemblies
- separation between the "topology" of the actor assemblies and their runtime semantics by picking alternative *Director* components

Passerelle/Ptolemy actors and sequences are a direct implementation of the well-documented *pipes-and-filters* architecture pattern, see for example [3]. Such an architecture promotes component-based solution designs with strong decoupling. Each actor in a complete process has a single responsibility that must be fulfilled based on data in messages received on the actor's input port(s). Results must be sent via the actor's output port(s).

Runtime semantics, or "models of computation", are determined through the usage of one of the "domains" provided by Ptolemy II. Passerelle is based on the *Process Networks* domain, which is a good basis for data-flow and process engines that internally require asynchronous behaviour and transparent actor-based concurrency.

Both actors and directors are implemented as Java classes, building on a rich API with base classes and utilities.

All of this results in a system that promotes designing and developing with reusability in mind. Actors are the basic level of reusable components and can be picked from actor libraries. There are actors for all basic flow control elements (branching, filtering, routing, loops, error handlers,...), but also libraries of domain-specific actors like Tango device control, database querying and updating etc.

Sub-processes can be easily stored in the library by end-users and can be reused between models.

Additionally, the clear split between actor development, solution assembly and execution environments supports different roles in solution delivery and maintenance for designers, developers, users and administrators.

ADVANCED CONCURRENCY FEATURES

Automation of non-trivial processes quickly confronts a solution developer with complex technical requirements. How can we control many simultaneous activities? How to assign system resources for a combination of long-running tasks and fast tasks? How to cater for peak loads in an optimal way?

The combination of asynchronous, event-driven internal processing, with the transparent concurrency that comes for free with the actor model means that neither the actor developer, nor the model designer, need to care too much about such technical complexities. Passerelle processes are automatically concurrent. Each actor can perform its work in an own thread that is managed internally by the engine. Each interaction between actors is buffered, so temporary overloads of an actor do not impact the others.

The philosophy behind the Passerelle engine, which is made possible thanks to the core Ptolemy design concepts, is that both an actor developer and a model designer should just focus on the desired functionalities. The complexities to ensure an optimal execution are hidden, and can be maintained and improved independently.

APPLICATION DOMAINS FOR SYNCHROTRONS

Passerelle actors and sequences can be used to automate all kinds of processes. By combining the right execution model with the right set of actors, the same process automation platform can handle e.g. :

- sequences for data acquisition and control
- workflows for data analysis
- autonomous monitoring and alarming
- scheduling batch processes
- …

A COMPLETELY MODULAR AND DYNAMIC PLATFORM

The Passerelle engine has been integrated in a complete platform for Java enterprise solutions with rich browser-based user interfaces. This has been designed from the ground up as a fundamentally modular system, through the usage of OSGi. This helps in several respects :

- strict control of intermodular dependencies with versioning
- OSGi bundles provide active modules with well-defined life-cycles
- support for dynamically updating/extending operational systems without downtime
- a very clean and performant service-based architecture

Just as for actors, OSGi promotes a model where each module has a limited responsibility, with a clean public interface and low coupling.

The Sherpabeans Java web framework integrates the following core technologies within an OSGi architecture :

- Apache Wicket for the web view layer
- JPA, Hibernate, eclipse link for persistence
- many smaller libraries

SherpaBeans itself is a collection of OSGi bundles that can be recombined at will. Some important modules are :

- scheduler for background jobs
- role-based security
- asset repository
- publish/subscribe notifications
- OSGi bundle upload and lifecycle management
- ...

This architecture makes it possible to deliver a basic Passerelle Manager platform with all core services, including a *web-based graphical model editor* (see Figure 1 below), an execution engine, some standard actor libraries and a complete web-based administration interface.

This system can then be extended in different ways by uploading extra modules as OSGi bundles (e.g. with new actors or new management features), designing and running new sequences etc.

ACTOR DEVELOPMENT WITH OSGI

In cases where the automated processes must be operational 24x7, even a short interruption to stop a sequence, and start a modified version, is problematic. Thanks to the dynamism offered by OSGi, many types of adaptations can be done without downtime.

A typical example is upgrading the logic of an actor implementation, e.g. for a bug fix or adapting to upgraded backend APIs etc.

To prepare for easy and dynamic maintainability, it is good practice to design actors in 2 layers :

- a service layer, containing the real logic behind a standardized task-based API that can be used by an actor
- a simplified actor, that basically feeds received data to the right service, collects its results and generates output messages from them

This results in actors that act as “binding” between a control layer (defined by the process model) and the service layer.

In an OSGi-based environment, each service can be offered by a dedicated bundle. When the bundle becomes active, the service is registered and is available for usage. When the bundle is deactivated or uninstalled, the service disappears. On the “client”-side, the actor can lookup the available service implementation(s), is notified about any changes in the status of the service, and can even pick the best one when multiple implementation versions would be available.

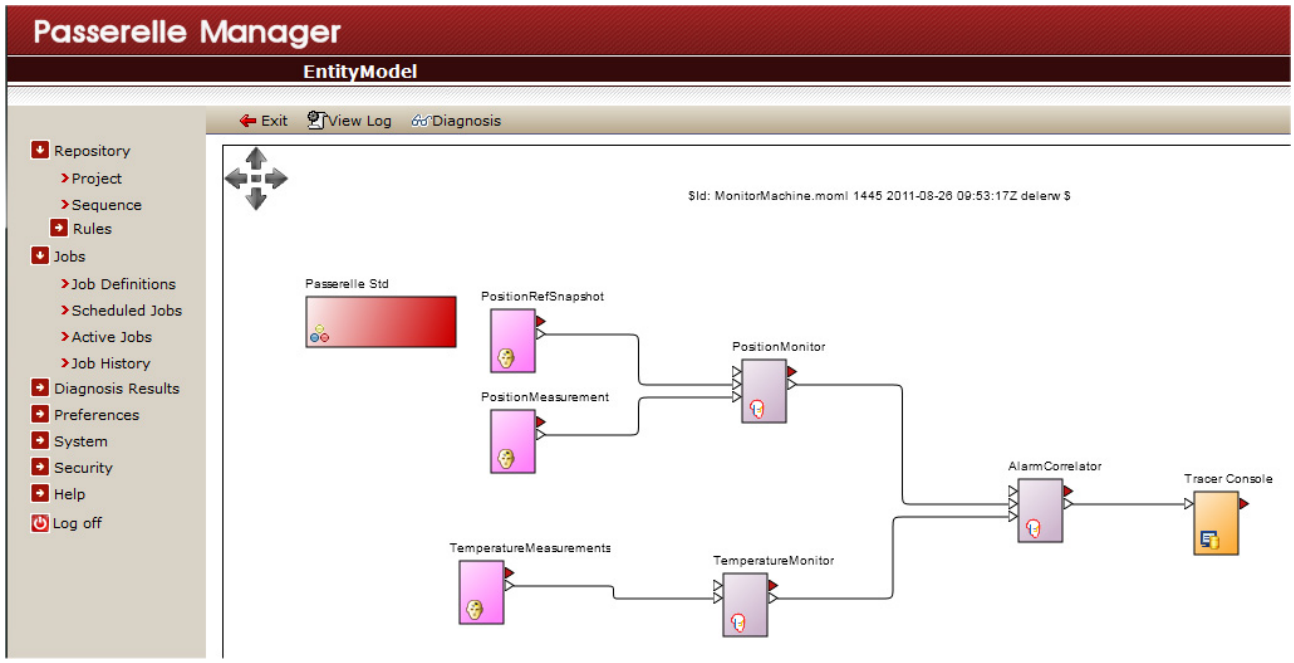


Figure 1: Screenshot from the web-based model editor in Passerelle Manager.

THE RESULT : EXTREME FLEXIBILITY FOR PROCESS AUTOMATION, WITHOUT DOWNTIME

Via the Passerelle Manager's web UI, it is possible to manage the lifecycle of each service bundle, to upload new versions etc.

The end result is a fully secured and dynamic platform to design new or updated processes via the browser, to upload new or updated actor and service implementations, to schedule process execution, consult execution traces etc.

COMBINING MODULES IN DIFFERENT PACKAGING OPTIONS

Through picking and recombining subsets of the available OSGi bundles, several solution packages can be obtained, with almost complete reuse of the core Passerelle bundles.

By using OSGi's service-based architecture, with a clear split between interface bundles and implementation bundles, it is also possible to provide several replaceable implementations of core service interfaces. For example, the asset repository has two implementations :

- the "enterprise" version, with persistence in a relational database, to be used inside the Passerelle Manager
- a simple file-based version, to be used with desktop workbench/IDE

Besides the Passerelle Manager, the following packages are possible :

- a standalone eclipse based Passerelle workbench
- the Passerelle eclipse workbench as plugins in larger workbench undertakings, like ESRF's Data Analysis WorkBench (DAWB) [4].
- an OSGi version with a Swing HMI (Figure 2 below)
- a plain Java-main Swing HMI, as OSGi bundles are finally also useable as plain jars...

EXAMPLE : ACTORS FOR DATA ACQUISITION VIA WEB SERVICES

In many enterprise environments, both scientific or non-scientific, the usage of web services is common, e.g. via SOAP or REST. Automated processes often include steps to obtain data from other software systems (often denoted as "backend systems") via such web services.

The preparation of a web-service request, and the interpretation of the received response typically involves tedious XML generation/parsing work. In many cases it's preferred to use frameworks like Apache Axis, JAXB etc to generate a Java binding.

In such a context, it is good practice to create an OSGi bundle for each required backend service. This bundle encapsulates the following responsibilities :

- accept request data from the client/actor
- prepare the outgoing request to the backend
- send the request, potentially with some safety measures to prevent overloading the backend
- collect the response or errors
- store the results and timing information etc in a database
- return the results to the client/actor

The actor layer becomes quite simple does not need to care about web service protocols, data persistence etc.

Such a design improves testability and maintainability.

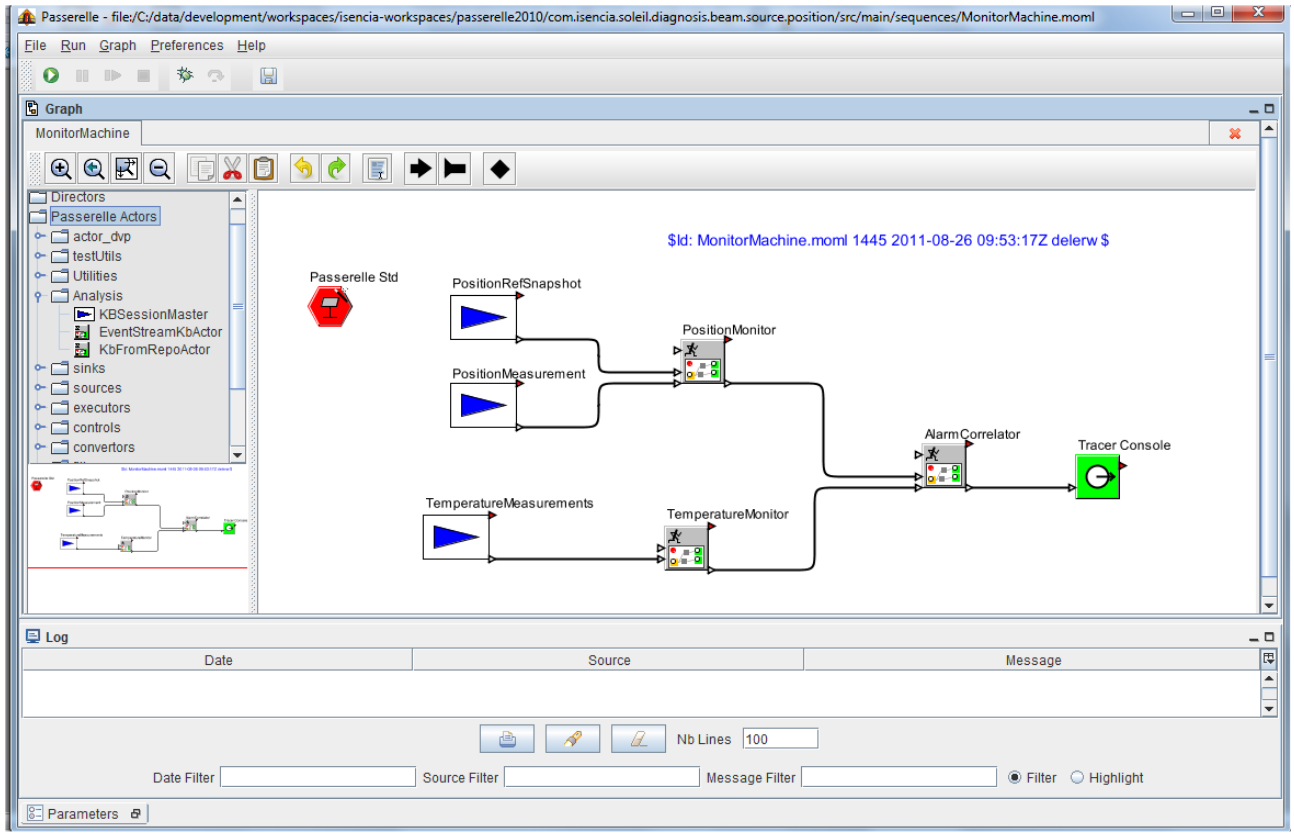


Figure 2: Screenshot from the Swing-based HMI/editor.

When the backend needs to change its web service, or the protocol changes for any other reason, or extra technical features like timeout management etc need to be added, this can all be done via an upgrade of the service bundle(s), without requiring a downtime.

CONCLUSIONS

Through a fundamental modular and componentized approach, it is possible to obtain advanced platforms for process automation.

By rigorously following good service-based design approaches in the context of an OSGi-based platform, a generic platform providing a complete tools set for designing, running and maintaining process models, consulting execution traces etc, becomes the starting point for a robust system that can be extended and updated without requiring downtime.

REFERENCES

[1] Passerelle project information : http://www.isencia.be/services/passerelle and http://code.google.com/a/eclipselabs.org/p/passerelle/

[2] Ptolemy II project information : http://ptolemy.berkeley.edu/ptolemyII/

[3] F. Buschmann et al., Pattern-oriented software architecture – Volume 1

[4] ESRF's DAWB project information : http://www.dawb.org

FPGA-BASED HARDWARE INSTRUMENTATION DEVELOPMENT ON MAST*

B.K. Huang, Dept. of Physics, Durham University, Durham and CCFE, Abingdon, Oxfordshire, UK
G. Naylor, N. Ben Ayed, G. Cunningham, A. Field, S. Khilar, CCFE, Abingdon, Oxfordshire, UK
R.M. Myers, R.M. Sharples, Dept. of Physics, Durham University, Durham, UK
R.G.L. Vann, York Plasma Institute, Dept. of Physics, University of York, Heslington, York, UK

Abstract

On MAST (the Mega Amp Spherical Tokamak) at Culham Centre for Fusion Energy some key control systems and diagnostics are being developed and upgraded with FPGA hardware: real-time data acquisition, vertical stabilisation, resonance mode driving, laser triggering, fast timer prototype, neutral beam monitoring and the bolometer upgrade. FPGAs provide many benefits including low latency and real-time digital signal processing. We accomplish lower costs by using industry standards such as the FMC (FPGA Mezzanine Card) Vita 57 standard [1] and by using COTS (Commercial Off The Shelf) components. Abstracting of the design and by using a flexible FPGA architecture gives resistance to obsolescence of modules and the ability to upgrade easily to add functionality by changing individual modules (connected by standard interfaces), avoiding the need to change the whole system.

FPGA DEVELOPMENT ON MAST

FPGAs are becoming more widely used in fusion research. In conjunction with appropriate hardware (DACs, ADCs, communication interfaces, etc) they offer a large number of capabilities including timing synchronisation, triggering, analogue/digital processing, I/O control, real-time feedback, embedded processors and more. As such this paper discusses this technology presented as a "standard architecture" where some systems (Resonance Mode Driving and Detection, Vertical Stabilisation, Real-Time Fast Acquisition, Fast Timer Prototype, NBI) use this architecture.

STANDARD ARCHITECTURE

MAST like other experimental facilities is a place where researchers are continually interested in investigating newly discovered phenomena. In order for researchers to be effective in this aim, it is necessary to have flexible hardware which allows researchers to adapt the system to the needs of their experiments. In many cases purchasing "turn-key" solutions to a cutting edge research experiment is either not possible or very expensive. We accomplish lower cost and flexibility instead by using a standard architecture that has the aim of being cost effective, flexible and reliable. By using open standards (Gigabit ethernet, FMC, SPI/IIC, AXI, ST Optical Fiber) it makes it easier to use different components of a system (such an FMC ADC board) in a variety of systems. This makes it easier to develop new systems, maintain and upgrade existing systems.

* This work was part-funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the Contract of Association between EURATOM and CCFE.

RESONANCE MODE DRIVING AND DETECTION

Toroidal Alfvén Eigenmodes (TAE) are global plasma modes at frequencies of 80 to 120 kHz. These eigenmodes can be considered analogous to orbit beam eigenmodes in synchrotrons and accelerators [2]. In MAST these modes can be driven by Neutral Beam Injection (NBI) or a time varying magnetic field generated by external coils.

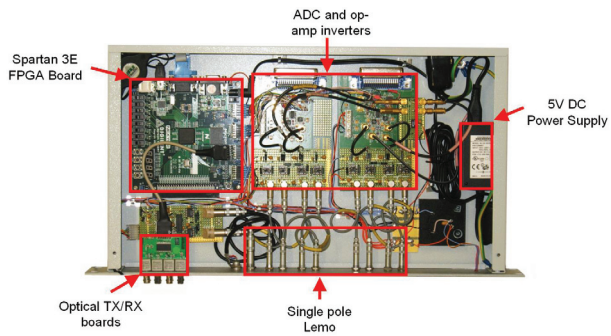


Figure 1: A resonance mode driving and detection system. Toroidal Alfvén Eigenmodes appear at ~80-120 kHz. An FPGA used system is used to create a time varying edge magnetic field perturbation by driving ~10 A sine wave in coils around the edge of the plasma.

The FPGA system (see Figure 1) is designed to drive these external coils. The FPGA creates an analogue sine wave via 16-bit DACs which are swept in frequency from 80 to 120 kHz. The sine wave is modified to have a fast zero crossing which reduces the reflected amplifier power. A ~10 A current is driven in the TAE coils using a matched resonant circuit. Parameters are loaded onto the FPGA using a serial UART connection which is connected to a rack-mounted serial over ethernet device. The benefit of UART is that it provides a low level communication (with a cost of only two wires) which can be used to debug the device. A future upgrade will likely switch this to gigabit ethernet. Additionally a 1 MHz 12-bit ADC can be used to digitise an arbitrary input signal for closed-loop real-time feedback, such as for tracking the resonant TAE frequency in real-time.

LASER TRIGGERING

A laser triggering system developed on MAST [3] is used to trigger 8 Nd:Yag Lasers 1.6 J 33 Hz (peak power 10 MW). The system incorporates a Spartan 3E FPGA with UART communication, fast rise-time triggers, and optical amplitude digitisation. This system is currently being enhanced for physics experiments in closed loop triggering applications to study plasma disruptions and instabilities.

VERTICAL STABILISATION

The Vertical Stabilisation system controls the vertical plasma position by adjusting the voltage in the P6 coils (see Figure 2).

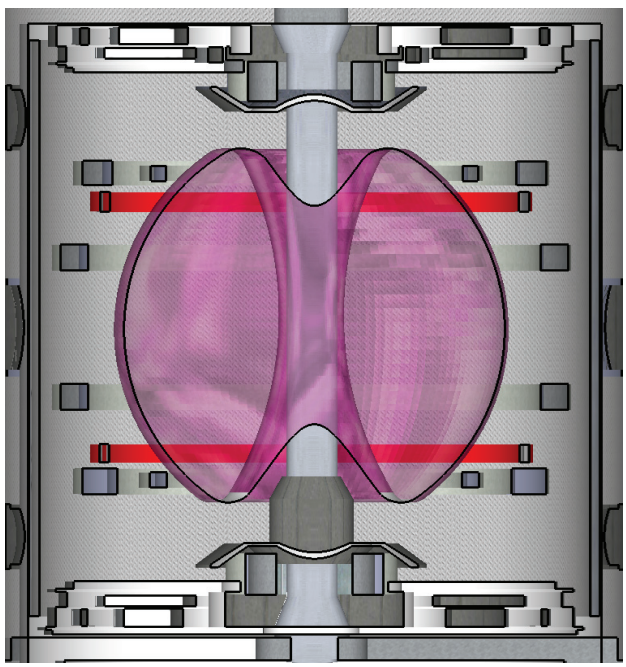


Figure 2: A cutaway view of the MAST vessel. The P6 magnetic coils which control the vertical position of the plasma are highlighted red. The coils are wired anti-parallel and driven by the vertical control system.

Currently an analogue vertical position control system is used to minimise the loop delay. However, there is little headroom between the optimal loop gains and those at which fast plasma events, such as 'Edge Localised Modes', will cause the amplifiers to trip. It is hoped that the use of an FPGA system (see Figure 3) will retain the benefit of low delay while allowing use of dynamic gain control and optimal filtering to make best use of the available dynamic range in the closed loop system.

The system comprises of: SP601 Spartan 6 FPGA, FMC12PEXP LPC Digital I/O, ADS1672 24-bit 625 ksps, remote management board, custom fibre optic boards for trigger/clocking, ATX power supply, 16-bit DAC for waveform generation.

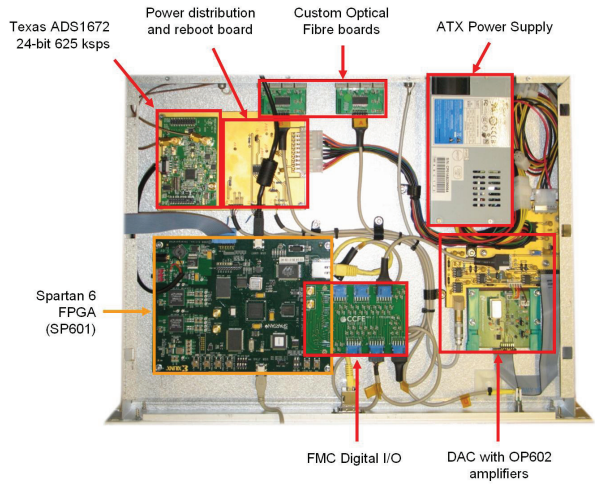


Figure 3: The new digital vertical stabilisation system contains a SP601 Spartan 6 FPGA. Gain coefficients and vertical position waveform are loadable over the Gigabit ethernet.

FAST REAL-TIME DATA ACQUISITION

The Microwave Imaging Diagnostic on MAST requires a fast real-time data acquisition system. Using the AXI bus it is possible to achieve a high continual streaming rate of 8 GB/s. This speed is based on 16 channels acquiring at 14-bits at 250 Msps: $16 * 2\,\text{Bytes} * 0.25\,\text{Gsps} = 8\,\text{GB/s}$. The system has been proved to be capable of achieving acquisition speed of >10 GB/s. This is compared to the theoretical rate of 12.8 GB/s for two DDR3 RAM chips 64-bits wide driven at 400 MHz DDR.

The hardware consists of 2xML605 FPGA Virtex6 boards, with on board 2 GB DDR3 RAM (4 GB Total), 2xFMC108 4DSP 14-bit 250 Msps ADC, FMC12PEXP LPC Digitial I/O, Gigabit Ethernet, Compact Flash, ATX Power Supply and Optical Reboot Board. The SSMC connectors on the FMC108 boards are internally wired to SMA front panel connectors. This is due to the fragility of SSMC (rated only for ~10 on/off connections) compared to the ~500 for SMA.

FAST TIMER PROTOTYPE

On experimental physics devices it is necessary to distribute a common clock to synchronise all instruments. A fast timer prototype (see Figure 4) has been developed to allow synchronisation using a distributed 10 MHz clock and generation of variable delay/length triggers. The system is a 1U system with ATX power supply, remote management board, LPC FMC12PEXP digital I/O, optical clock and trigger boards. The system uses an SP601 Spartan 6 board with Gigabit ethernet for communications. The generated pulse has a 37 ns precision with ~4 ns jitter. On board parallel (BPI) flash to store the firmware to makes this system fully standalone (the network is not needed for booting).

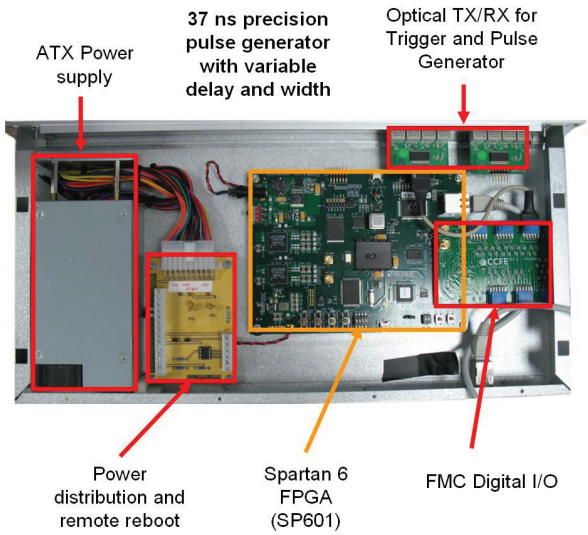


Figure 4: The Fast Timer Prototype. A simple system that contains many aspects of the FPGA standard architecture showing the ease with which new diagnostics and control systems can be developed. It is a completely remote managed system that communicates over ModBus.

NBI FPGA DIAGNOSTIC

The Neutral Beam Injection (NBI) system on MAST injects over 3.2 MW of power into the tokamak. As part of the running operations there are occasionally arcing currents. Currently these are monitored at 20 kHz sampling rates but since arcs can happen over smaller timescales than 50µs. This new system (see Figure 5) will include: 4 Channel 1 Msps 12-bit ADC, ATLYS Spartan 6 FPGA, ATX power supply, remote reboot board.

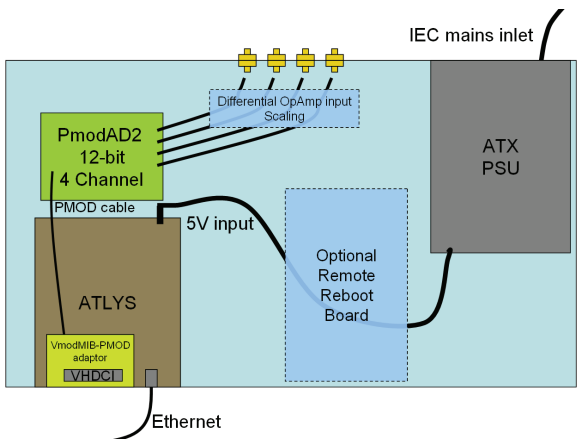


Figure 5: The NBI arc detection acquisition system. This system increases the sampling rate from 20 kHz to 1 MHz in order to better resolve the voltage waveform during an NBI arc event. This system utilises an ATLYS Spartan 6 board and follows the FPGA standard architecture.

BOLOMETER UPGRADE

The bolometer array is a 32 channel system on MAST which measures plasma irradiation from visible to soft X-ray. The detection system is based on gold foil configured as a wheatstone bridge whereby incident radiation causes a voltage difference across the bridge.

The challenge with this system is that in any given time interval a very small amount of power falls onto the gold foil thus making the diagnostic sensitive to noise. An additional requirement is low temperature drift. In order to increase the sensitivity of the system and help prevent problems with low DC offsets and 1/f noise a chopping sine wave at 20 kHz is used. The hardware will be FPGA based to control the calibration of the wheatstone bridge, digital filtering, control waveform and other control operations.

CONCLUSIONS

Significant FPGA development is occurring on MAST in key control systems using a standard FPGA architecture. The adoption of a standard architecture has enabled the development of a number of different diagnostics performing a variety of different functions.

ACKNOWLEDGEMENTS

This work was funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the contract of Association between EURATOM and CCFE. The views and opinions expressed herein do not necessarily reflect those of the European Commission. Our thanks to the MAST Team who have helped in the development of these FPGA systems.

REFERENCES

[1] P.R. Alvarez, M. Cattin, J. Lewis, J.Serrano, T. Włostowski, "FPGA mezzanine cards for CERNs accelerator control system", `oai:cds.cern.ch:1215572`, Technical Report CERN-ATS-2009-097, CERN, Geneva, Nov 2009.

[2] Y. Tian, L.-H. Yu, "NSLS-II Fast Orbit Feedback with Individual Eigenmode Compensation", WEODN4, Proceedings of PAC2011, New York, USA, 2011, `http://JACoW.org`.

[3] G.A. Naylor, "An FPGA based control unit for synchronization of laser Thomson scattering measurements to plasma events on MAST", *Fusion Engineering and Design*, 85(3-4):280-285, 2010. Proceedings of the 7th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

FPGA COMMUNICATIONS BASED ON GIGABIT ETHERNET

L.R. Doolittle, C. Serrano, LBNL, Berkeley, CA 94720, USA

Abstract

The use of Field Programmable Gate Arrays (FPGAs) in accelerators is widespread due to their flexibility, performance, and affordability. Whether they are used for fast feedback systems, data acquisition, fast communications using custom protocols, or any other application, there is a need for the end-user and the global control software to access FPGA features using a commodity computer. The choice of communication standards that can be used to interface to a FPGA board is wide, however there is one that stands out for its maturity, basis in standards, performance, and hardware support: Gigabit Ethernet. In the context of accelerators it is desirable to have highly reliable, portable, and flexible solutions. We have therefore developed a chip- and board-independent FPGA design which implements the Gigabit Ethernet (GbE) standard. Our design has been configured for use with multiple projects, supports full line-rate traffic, and communicates with any other device implementing the same well-established protocol, easily supported by any modern workstation or controls computer.

INTRODUCTION

The FPGA communication system based in GbE is the result of two success stories. First FPGAs, whose capabilities have improved dramatically in the last few years in terms of size, performance, and flexibility. This tremendous improvement has enabled the range of applications of FPGA solutions to explode, increasing the volume of production, and therefore reducing the price of every new generation of FPGA family appearing on the market. The second success story corresponds to Ethernet (and GbE as the highest performance flavor to see adoption in today's commodity hardware). The key to the success of Ethernet resides in its simplicity and its associated side effects [1]. Implementation is simple in hardware and provides larger data rates than most computers can process these days. The evolution of the Ethernet protocol since its creation has accompanied that of computers since its creation in 1970s, and has been the Local Area Network (LAN) protocol by excellence ever since (with the market of Ethernet switches alone amounting to $16 billion in 2010).

In accelerators, FPGAs have introduced a great deal of flexibility and performance into machine fast controls and operations. They are extensively used to implement highly specialized tasks where simplicity, low production cost, and reliability are big assets. Even if simple and reliable, these systems need means for communications with commodity computers for configuration, data acquisition and diagnostics via Global Controls such as EPICS and TANGO. The nature of highly specialized hardware, combined with the simplicity of the GbE protocol make the implementation of GbE in FPGAs our preferred way of communication with the Global Controls.

STANDARD SELECTION

Once computers are taken away from highly specialized hardware in accelerators, a number of hardware standards other than GbE are available to enable communication between FPGAs and Global Controls, such as: PCI, USB, CAN, VME, VXI, cPCI, PCIe, IEEE-488 (HP-IB), IEEE-1392 (Firewire), SATA (or eSATA), etc. Longevity is a usual requirement for hardware installed in accelerators, and the chosen communication standard should last at least as long as the actual hardware. While CAN, VME, IEEE-488 and cPCI are indeed standards, they do not achieve as much performance/simplicity ratio as that provided by GbE. PCI hardware will undoubtedly be present for a long time, however active development seems to have shifted from the original parallel standards to the high-speed serial PCIe version.

USB can be amazingly attractive for small projects. Throughput is high, as is the availability for hardware components for both sides of the link. On the other hand, both software and hardware models are very much desktop-based. Cable distance is limited, and the standard is young and unstable enough that software support can be quirky. The LBNL experience running USB-based instrumentation affirms that it should be avoided for production accelerator installations. Ethernet on the other hand accounts for maturity, stability, high availability of hardware, high performance and ease of integration with highly specialized hardware in accelerators.

OVERVIEW

In addition to GbE, our implementation includes IP, UDP and ARP at full line speed. UDP seems a better match to real-time communications than TCP since point-to-point dedicated networks are normally available in accelerator deployments, and is much better suited to the finite resources of an FPGA. The additional complexity added by supporting UDP over raw Ethernet is largely rewarded by the extensive UDP support on the host side (BSD sockets API, and its descendants). Raw Ethernet support is not as usual, and programs manipulating raw Ethernet packets typically require elevated levels of privileges, removing one layer of security.

Fig. 1 shows a block diagram of an implementation example using the FPGA Ethernet module. The core func-

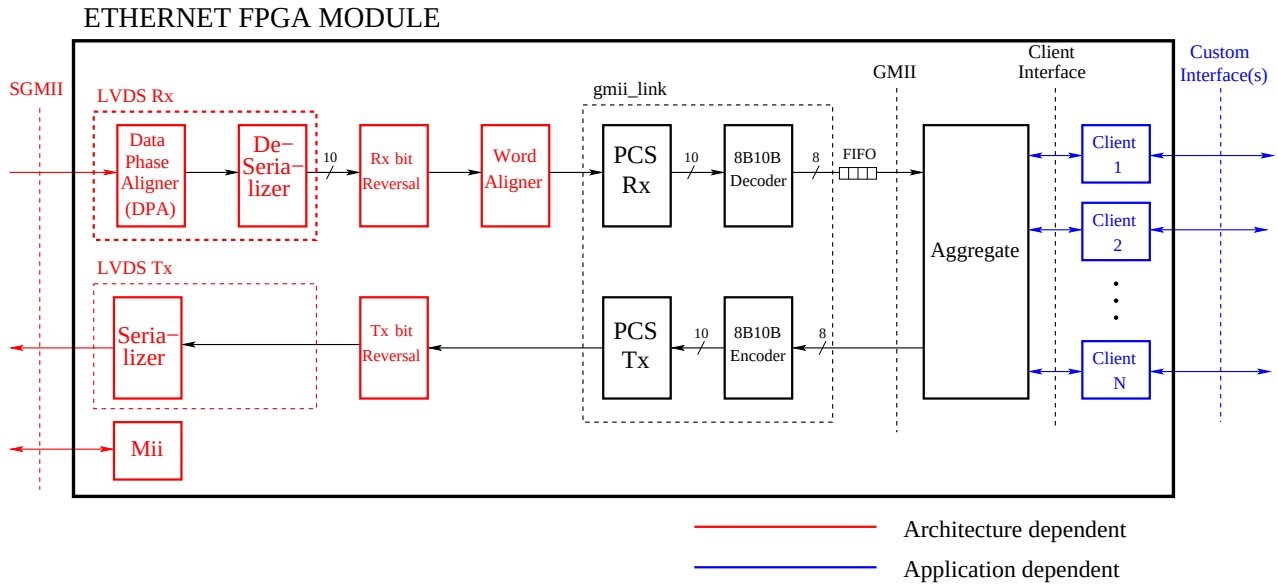


Figure 1: Top level block diagram. Example where the architecture dependent modules correspond to the implementation on an Altera Stratix-IV EP4SGX230KF40C2ES FPGA.

tionality is implemented in the Aggregate module, which has been custom-designed using Verilog HDL. A clean separation between this module and rest of the design allows for flexibility and portability, benefiting from a well defined medium-independent interface in the GbE side (GMII in the example shown in Fig. 1, which is defined as part of the GbE standard), and another well defined client interface for data exchange with the FPGA fabric. Details on the core design and interfaces are given next.

CORE DESIGN

The core design described here includes the architecture/implementation independent modules, shown in black in Fig. 1. This logic is independent of the targeted FPGA (as well as any peripherals), the particular application of the communication, and implements layers 2, 3, and 4 of the OSI model.

The FPGA Ethernet module has compile-time configurable IP and MAC addresses, and the current implementation assumes that the host side uses a single UDP port number for communication. Aggregate implements the upper level logic to (de)encapsulate UDP packets. It extracts the UDP packet length in the length field and uses it to generate the necessary control signals needed to interact with the client modules in the FPGA fabric. The host has the possibility of communicating with any of the n clients (in blue in Fig. 1), where each client is assigned a UDP port number. More details on the clients are given later.

The PCS Rx/Tx modules implement the 802.1x PCS standard, building a block which interfaces the Ethernet framer to Tx PMA (Physical Medium Attachment). It performs preamble generation, inserting idle patterns, 802.1x link negotiation and all the low-level signaling (excluding 8B10B coding, which is implemented in the 8B10B Encoder/Decoder). The 32-bit CRC of the Ethernet frame, and the length of the IP and UDP packets are checked for correctness. If any errors are detected, the whole Ethernet frame is dropped and the clients stay idle. Details on the clients are given next.

INTERFACES

Client Interface (FPGA side)

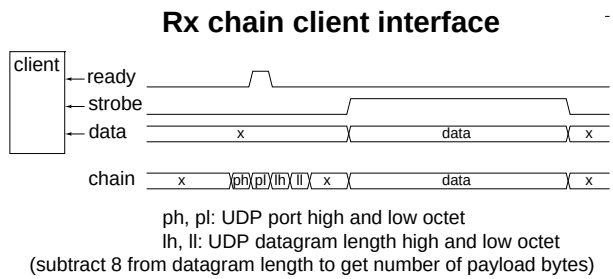


Figure 2: Rx chain in the client interface shown in Fig. 1.

The Aggregate module provides the data contained in the UDP packet payload to the client modules (application dependent modules shown in blue in Fig. 1), where the distinction of each client is done using the UDP port number. These clients exist to provide the flexibility of implementing different custom protocols adapted to the application needs. This level of communication relies on data transaction from levels 1 through 4 in the OSI model, and custom interfaces to interact with the rest of the fabric in the FPGA. The different client instantiations share a common interface with the Aggregate module (labeled “client interface” in Fig. 1), and each implement a custom interface

with the rest of the fabric in the FPGA (labeled "Custom Interface(s)" in Fig. 1).

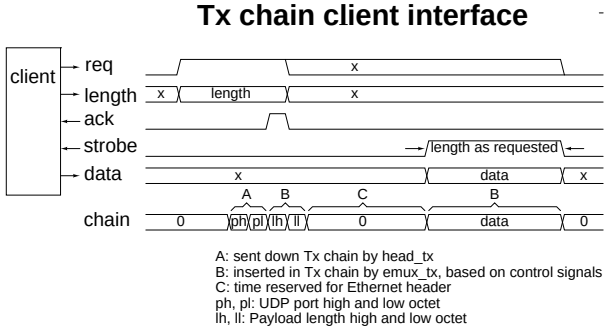


Figure 3: Tx chain in the client interface shown in Fig. 1.

Figs. 2 and 3 show the Rx and Tx flow chain for the client interface, respectively (seen from the clients' perspective). The Rx chain follows a simple strobe and data transmission, where the strobe is high if the data is valid, and the UDP packet it was sent into contained the UDP port associated with the client. A ready signal is provided a fixed number of clock cycles before the strobe, which can be used by the client module to anticipate when it will receive valid data.

The Tx chain follows a similar transmission scheme (strobe and data), preceded by a transmission request for a scheduler to assign when each client transmits the data onto the Ethernet link. When the client is ready to send data, it sets the request signal along with the length of the data to be transmitted. Once the scheduler has established the turn of the client to send the data, it sends an acknowledge to the client, that will then receive a strobe for as many clock cycles as previously specified in the length field.

The design is modular, and each UDP clients is handled by a different Verilog module. In addition to that, the project package includes means to automatically generate Verilog code for the Aggregate module, including automatic replication of client handling logic, and the proper module prototype to connect to as many clients as needed.

On-board Local Bus to UDP Gateway

Local bus encoded 64−bit frame

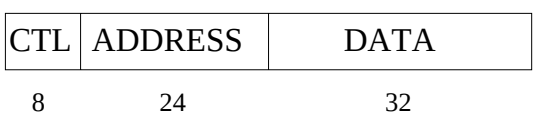


Figure 4: Local bus 64-bit frame.

In the general case, the Ethernet module provides a link between UDP/IP, and a series of customized interfaces, There are as many clients as protocols are encoded in the UDP packet payloads, and each client is assigned a UDP port number to be accessed from the host side. One example of client is what we call the Local Bus to UDP Gateway, where the access to FPGA registers is provided by a very simplified on-board data, address, strobe type local bus.

The master of the transaction is the software side, which can send series of register read and write commands to the FPGA by concatenating the 64-bit frame pattern (shown in Fig. 4) into the UDP payload. One of the 8 control bits is used to indicate the read/write command. If a read command is found, the FPGA ignores the data field and responds using the same 64-bit pattern, where the control and address field remain unchanged, and the data field is filled with the corresponding register value. On the other hand, if a write command is found, the data field is written onto the corresponding register, and the same frame is sent back to the host (which can use it as an acknowledgement). This scheme described above has an intrinsic bandwidth limit for the FPGA, since it only sends data upon request.

Host Interface (GbE link side)

The Ethernet protocol has increasingly added support for different data rates and physical media, and different FPGAs implement the physical layer (including serializer and deserializer mechanisms) differently. It was therefore indicated to provide a medium-independent interface to the core of the design in order to preserve generality and portability. A GMII interface (defined as part of the Ethernet standard) is provided to connect the FPGA core logic and the architecture specific modules (in red in Fig. 1).

The package described here also includes means to attach a live simulation of the synthesizable Verilog to a Linux tun/tap interface, where the GbE link can be abstracted and third party software can be used to exchange UDP packages with the FPGA logic, which has been extremely helpful in the debug process and can be used for early development of software without the need of actual hardware.

CONCLUSIONS

The Ethernet module has been used in various projects under different inter-lab collaborations. The package contains (not counting various instantiation of hardware primitives) purely synthesizable Verilog. There is no need for off-chip memory and synthesizes using 930 logic cells, and 3 block RAMs on a Xilinx Spartan-6 XC6SLX45T (which accounts for 1.7% and 2.6% of the available resources respectively). The physical layer has been demonstrated functional using GMII, Xilinx Virtex-5 MGT, and Altera Stratix-IV LVDS. The project is continuously evolving as new needs arise, and physical layer support plans or currently ongoing testing include RGMII (double-date-rate GMII), Spartan-6, and Virtex-6.

REFERENCES

[1] Rich Seifert, "Gigabit Ethernet: Technology and Applications for High Speed LANs", Addison-Wesley, 1998

THE UPGRADE PATH FROM LEGACY VME TO VXS DUAL STAR CONNECTIVITY FOR LARGE SCALE DATA ACQUISITION AND TRIGGER SYSTEMS

Chris Cuevas, David Abbott, Fernando Barbosa, Hai Dong, William Gu, Edward Jastrzembski, Scott Kaneta, B. Moffit, Nick Nganga, Ben Raydo, Alexander Somov, William Mark Taylor, Jeff Wilson, Thomas Jefferson National Accelerator Facility, Newport News, Virginia, U.S.A

Abstract

New instrumentation modules have been designed by Jefferson Lab that take advantage of the higher performance and elegant backplane connectivity of the VITA 41 standard of VXS. These new modules are required to meet the 200 KHz trigger rates envisioned for the 12GeV experimental program. Upgrading legacy VME designs to the high speed gigabit serial extensions that VXS offers, comes with significant challenges, including electronic engineering design, plus firmware and software development issues. This paper will detail our system design approach including the critical system requirement stages, and explain the pipeline design techniques and selection criteria for the FPGA that require embedded Gigabit serial transceivers. The entire trigger system operates synchronously at 250MHz, utilizing global clock and synchronization signals distributed to each front-end readout crate where finally the VXS switch slot distributes these signals to all front-end modules. The readout of the buffered detector signals relies on 2eSST over the standard VME64x path at >200MB/s. We have achieved an aggregate of 64Gb/s rate of trigger information from payload to switch slots within one VXS crate and will present results using production modules in a two crate test configuration with both VXS crates fully populated. The VXS trigger modules that reside in the front end crates, will be ready for production orders by the end of the fiscal year. VXS Global trigger modules are in the design stage now, and will be complete to meet the installation schedule for the 12GeV Physics program.

VXS SELECTION CRITERIA

VME with serial extensions or VXS was selected as the 12GeV data acquisition backplane foundation for the front end detector readout and trigger hardware interface. The 6GeV experimental program relied on FastBus ADC and TDC with VME single board computers providing the configuration and readout of Physics data, along with custom electronics developed for the trigger system. [1] The new requirements for 12GeV experiments demand higher trigger and data readout rates as shown in Table 1.

We have a successful record with VME and VME64x custom data acquisition and trigger modules that were developed during the 6GeV program to replace aging and obsolete hardware. When VITA41 (VXS) [2] was initially emerging we decided that the specifications for the new serial extensions would meet and exceed our requirements for creating digital energy summation for hundreds of ADC channels within a VXS card enclosure. The VXS backplane also is an elegant solution to distribute critical and essential global clock and synchronization signals to each front end module without cumbersome cabling or additional rear transition hardware. The VXS selection offered low risk and allowed a logical upgrade path for our legacy hardware. The VME consortium was promoting the addition of gigabit connectivity to a very large customer base, and the new VXS standard remains a viable choice for the future. [3]

Table 1: Experiment Rates and L1 Trigger Channel Count.

	Hall B CLAS12	Hall D GlueX
L1 Trigger Channels	4,000	4,500
#VXS Crates (Total Crates)	38 (43)	25 (52)
#Single Board CPU	43	52
L1 Trigger Rate	<20KHz	<200KHz
L1 Data Rate(MB/s)	60MB/s	3GB/s
L3 Farm Data Rate (Disk MB/s)	10KHz (60MB/s)	20KHz (300MB/s)

SYSTEM TOPOLOGY OVERVIEW

VXS offers dual star switch capability with 18 payload slots connected to two central switch slots. The VITA 41.0 specification lists 4 full duplex gigabit connections or "lanes" from each payload slot, to the each switch slot and a line rate of 10 Gb/s, so 80 Gb/s bandwidth is offered with dual star connectivity. At the start of our R&D hardware design phase in 2005, we were using the Xilinx Virtex-IV series with gigabit transceivers capable of 3.125Gb/s. Our production payload modules will use Xilinx Virtex-V transceivers capable of 6.25Gb/s line speeds. All four full duplex lanes are routed on the circuit board, but presently only two lanes are used to one switch slot. The front end payload modules are sixteen channel, 250MHz flash ADC boards [4] that use these gigabit lanes to transmit digital sum information to one of the VXS switch slots.[5] This switch slot module is a Crate Trigger Processor (CTP) which collects 64Gb/s from the 16 payload modules and processes a crate summation that is optically transmitted over a parallel fibre optic cable[6] to a Sub-System Processors (SSP) and finally to a Global Trigger Processor (GTP)[7] that determines if a Physics trigger condition was satisfied.

HARDWARE DESIGN CHALLENGES

High Speed Gigabit Serial Transmission

The design challenges for reliable and successful transmission of gigabit serial data over the VXS backplane requires the investment of high speed circuit board layout and routing tools. The FPGA selection requirements include at least four full duplex Gigabit Transceivers, user I/O pin count >500, and fast integrated block memory with multi-rate FIFO logic. We use circuit board routing simulation tools such as Mentor Graphics HyperLynx [8] which are invaluable for critical simulation and verification of circuit board signal integrity for the gigabit transmission paths before the manufacturing process. The FPGA devices that we use are capable of 6.25Gb/s serial transfer, and we have designed our circuit boards with signal integrity techniques using standard FR4 circuit board material to achieve >2.5Gb/s which meets the data transfer bandwidth requirements.

Another significant investment required for the hardware verification of the gigabit transceivers was a digital signal analyzer with 8GHz bandwidth to measure and record the backplane and fibre optic gigabit transceiver performance and to perform jitter analysis on the critical system clock and synchronization signals with at least 1ps resolution. We have purchased the Tektronix jitter analysis software which is a critical tool for the verification of our system clock, and for measurements of the phase controlled jitter attenuated clock provided by the Signal Distribution (SD) switch card in every crate [9].

The investment of firmware development tools from FPGA industry leaders, Xilinx and Altera were also taken into consideration for the upgrade path to VXS, and costs for firmware simulation and verification tools from industry leading vendors must be considered because the license and maintenance fees are not trivial.

We use the Xilinx Aurora protocol for serial transmission which is robust, simple and is included with the FPGA development tools. Considerations for forward error correcting algorithms have been planned, but not implemented at this time.

Fiber Optic Distribution Network

As shown in Figure 1, the digital sum value from each CTP in the front-end crate, and the distribution of the global clock, synchronization and trigger commands from the global trigger hardware, use a separate fibre optic cable. The crate sum fibre link is shown in orange, and the critical timing signals distributed to each front-end crate are blue. Each fibre optic link makes use of the Avago POP4 fibre optic transceivers and parallel OM3 rated glass fibre cable with MTP connections. These fibre optic transceivers operate at 3.125 Gb/s for an aggregate bandwidth of 10 Gb/s which is ample enough for the summing information that is sent forward to the global trigger processing hardware. The fibre link used for the distribution of the global clock, critical timing signals and trigger commands run at 1.25 Gb/s.

Global Trigger Essentials

The global trigger SSP modules collect the 10Gb/s trigger information streams from each of the front end crates via the parallel fibre optic connections. These trigger data streams represent energy sums from calorimeter detector apparatus and hit patterns from Time-Of-Flight (TOF) detectors. The SSP are VXS payload modules, so after the SSP has performed the required sub-system alignment, the trigger data is passed forward to the GTP on the VXS backplane. The final trigger word is cabled to the Trigger Supervisor (TS) [10] and is distributed to the front-end crates and modules to initiate data readout.

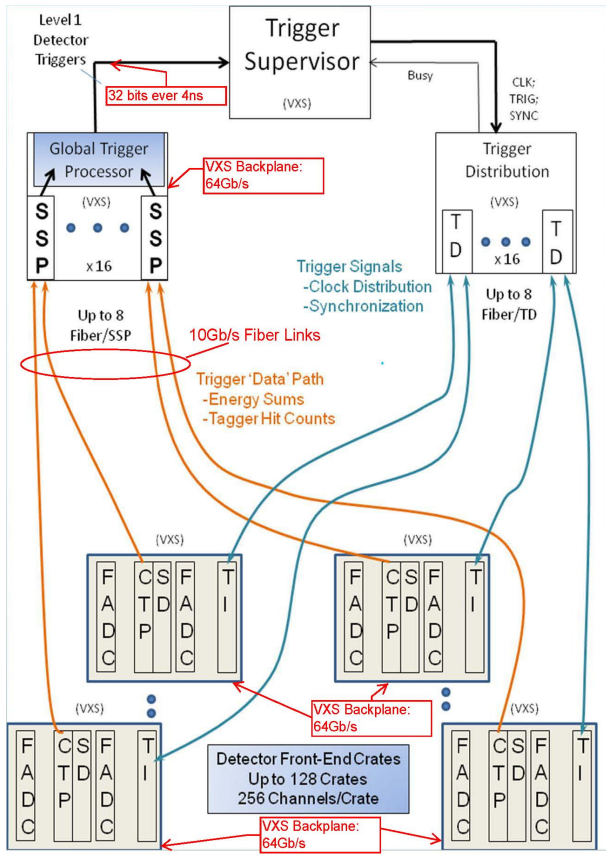


Figure 1: Gigabit Serial Links.

System Latency and Transport Protocol

The Level 1 trigger is formed from the detector signals that have been selected for a given Physics event. The energy sums from a calorimeter, for instance, are created in firmware and transported on the VXS backplane using Xilinx's Aurora™ [11] protocol. Aurora™ is the low latency communication protocol between FPGAs on payload and switch boards. Other serial gigabit protocols such as Ethernet or PCIe cannot be used because the latency is too large to meet the 3.2μS requirement. This requirement comes from our choice of a commercial

Time-Digital-Converter ASIC [12]. These protocols include other properties that are not required for the transfer of the digital sum information and will add significant overhead compared to Aurora™. The Aurora™ protocol is also used on the fibre optic communication interface, and Aurora provides full duplex lane bonding to achieve aggregate bandwidth sufficient to transfer information from each crate to the global trigger crate modules.

Data Rates with Linux Single Board Computers

The VXS crates use single board computers (SBC) and the Data Acquisition Group at Jefferson Lab has evaluated the performance offered by various companies.

In our present two crate test station, one SBC is a GE-7875 (Intel Core 2 Duo) and the other VXS crate uses a Concurrent v717 (Intel mobile core i7). These SBC have similar interrupt response times of ~5μs and maximum network throughput of 116MB/s using only 5% of the CPU resource. These SBC support 2eSST VME transfer mode and with a 200 KHz trigger rate, we achieve a nominal VME backplane transfer of >40MB/s. Each crate can accept 256 coaxial detector signals, and typical occupancy for a fully populated crate will be <10%.

LATEST PERFORMANCE RESULTS

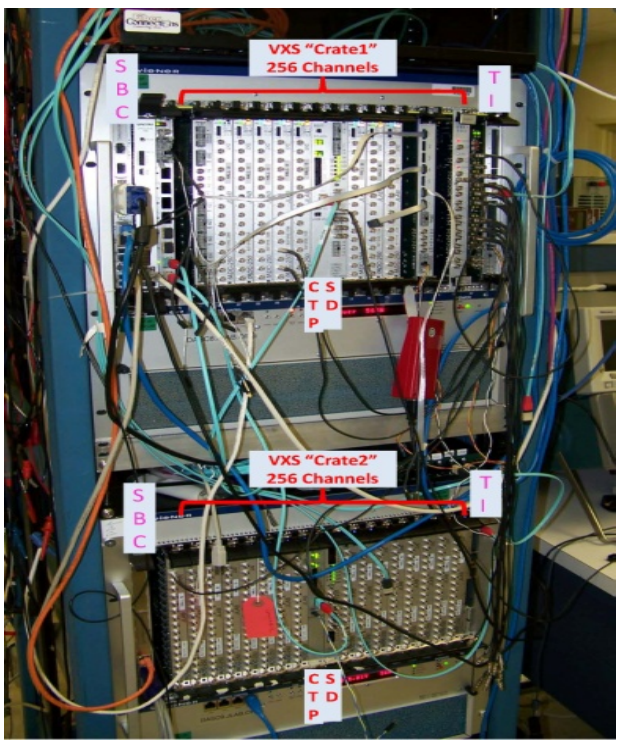


Figure 2: Two VXS crate test station.

A photo of two VXS crates, which are almost fully populated with payload boards, is shown in Figure 2.

Each payload module has 16 coaxial inputs, and these inputs are captured with a 12-bit flash ADC. A fully populated crate can accept 256 coaxial detector signals and all channels participate in creating an energy sum value which is transferred every 4ns. Depending on the location of the detector apparatus, the typical number of channels for a Physics trigger that will have relevant signal occupancy is <10%. For our two crate testing, we inject pulse signals to at least 12% of the channels to simulate a typical occupancy from a detector apparatus.

Figure 3 shows data transfer rate versus a Level 1 Trigger rate with channel occupancy at 12.5%. The aggregate data transfer rate of >40MB/s is from both VXS crate SBC. Selected input channels produce pulses at trigger rates set with a programmable generator, and several trigger rates points are shown. The Trigger Interface has the capability to store up to 256 trigger events (block) before signalling the SBC for VME 2eSST block readout. This block-event capability allows the fully pipelined system to accept a 200 KHz trigger rate and transfer the Physics event data from the SBC to the software trigger storage server. Figure 3 also shows the advantage of event blocking and larger events per block reduce CPU response time and data overhead per event.

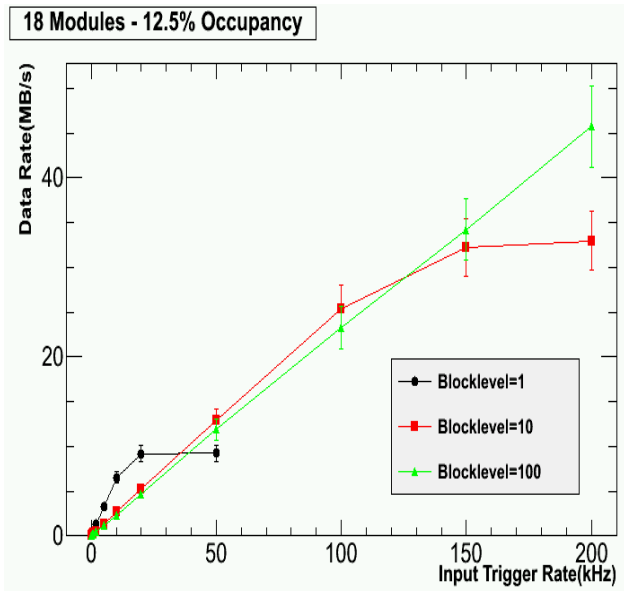


Figure 3: Trigger rate vs. Data rate and Block-level affects.

CONCLUSION

We have met the requirements and milestones established for the new front-end data acquisition hardware and trigger system functions defined at the conceptual design phase for the 12GeV experimental upgrade project.

The upgrade path the VITA 41 or 'VXS' presented many significant engineering design challenges for both circuit board and firmware development. Careful planning including the cost of essential circuit board design and simulation tools, plus the required test equipment is critical for success. We now have the essential experience with high speed gigabit serial backplanes, FPGA transceiver technology, and firmware development.

In the next two years we will be challenged with production testing of the new front-end and trigger

hardware, and will be required to meet the implementation goals for pre-commissioning the detector readout and trigger systems in the 12GeV experimental areas. The two crate VXS test station has been an excellent foundation for the development of data acquisition and trigger commissioning 'tools' that will be essential for full qualification of the VXS hardware systems needed when the 12GeV Physics programs begin.

ACKNOWLEDGEMENT

Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177 The U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce this manuscript for U.S. Government purposes.

REFERENCES

[1] CLAS-Note 91-017; "A VXI Based Trigger for the CLAS Detector at CEBAF" D. Doughty Jr. J. Englert, R. Hale, S. Lemon, CNU; C. Cuevas, D. Joyce, CEBAF 1991.

[2] ANSI/VITA 41.0-2006; VMEbus Switched Serial Standard, (2006)

[3] VME and Critical Systems Magazine; "Mil tech gets smart with OpenVPX, VXS designs"; May, 2011 p. 30-31

[5] "Integrated Tests of a High Speed VXS Switch Card and 250Msps Flash ADCs", H. Dong, C. Cuevas, D. Curry, E. Jastrzembski, F. Barbosa, J. Wilson, M. Taylor, B. Raydo. *IEEE Nuclear Science Symposium* N15-375, 2007.

[6] 2010 Panduit Corporation; "Panduit® QuickNet™ MTP Trunk Cable Assemblies" http://www.panduit.com

[7] "The Global Trigger Processor: A VXS Switch Module for Triggering Large Scale Data Acquisition Systems"; S.R. Kaneta, C. Cuevas, H. Dong, W. Gu, E. Jastrzembski, N. Nganga, B.J. Raydo, J. Wilson JLAB, Newport News, Virginia, USA; *ICALEPS 2011- WEPMS017*

[8] 2010 Mentor Graphics "HyperLynx®" Signal Integrity Tools and Circuit Board Verification"; http://www.mentor.com

[9] "Delivering Phase Controlled Jitter Attenuated Clock Signals to Data Acquisition System"; ı *IEEE Nuclear Science Symposium* NP2.S-185; 2011

[10] Jefferson Lab – September, 2011; "The Trigger and Clock Distribution for 12GeV Upgrade Experiments"; DAQ Group and Fast Electronics Group

[11] Xilinx SP002 "Aurora v2.4 Protocol Specification"; January 10, 2006: http://www.xilinx.com/aurora

[12] "F1: An Eight Channel Time-to-Digital Converter Chip for High Rate Experiments"; Braun, G ; Fischer, H ; Franz, J ; Grünemaier, A ; Heinsius, F H ; Hennig, L ; Königsmann, K C ; Niebuhr, M ; Schierloh, M ; Schmidt, T *et al.* 5th Conference on Electronics for LHC Experiments, Snowmass, CO, USA, 20 - 24 Sep 1999, pp.383-387F1TDC

OPEN HARDWARE FOR CERN'S ACCELERATOR CONTROL SYSTEMS

Erik van der Bij#, Pablo Alvarez Sanchez, Myriam Ayass, Andrea Boccardi,
Matthieu Cattin, Carlos Gil Soriano, Evangelia Gousiou, Samuel Iglesias Gonsalvez,
Gonzalo Penacoba Fernandez, Javier Serrano, Nicolas Voumard, Tomasz Wlostowski,
CERN, Geneva, Switzerland

Abstract

The accelerator control systems at CERN will be renovated and many electronics modules will be redesigned as the modules they will replace cannot be bought anymore or use obsolete components. The modules used in the control systems are diverse: analog and digital I/O, level converters and repeaters, serial links and timing modules. Overall around 120 modules are supported that are used in systems such as beam instrumentation, cryogenics and power converters. Only a small percentage of the currently used modules are commercially available, while most of them had been specifically designed at CERN.

The new developments are based on VITA and PCI-SIG standards such as FMC (FPGA Mezzanine Card), PCI Express and VME64x using transition modules. As system-on-chip interconnect, the public domain Wishbone specification is used.

For the renovation, it is considered imperative to have for each board access to the full hardware design and its firmware so that problems could quickly be resolved by CERN engineers or its collaborators.

To attract other partners, that are not necessarily part of the existing networks of particle physics, the new projects are developed in a fully 'Open' fashion. This allows for strong collaborations that will result in better and reusable designs.

Within this Open Hardware project new ways of working with industry are being tested with the aim to prove that there is no contradiction between commercial off-the-shelf products and openness and that industry can be involved at all stages, from design to production and support.

OVERVIEW OF CONTROLS HARDWARE

The Controls Group of the Beams Department (BE-CO) at CERN is responsible for the specification, design, procurement, integration, installation, commissioning and operation of the controls infrastructure for all CERN accelerators, their transfer lines and the experimental areas. The group provides services like general machine and beam synchronous timing generation and distribution (see Figure 1) and signal observation systems, as well as support for drivers and higher-level software.

As basis the group uses a set of standardized hardware and software controls components. The hardware used in the control systems is diverse: analog and digital I/O, level converters and repeaters, serial links and a range of timing modules. Overall around 120 modules are supported that are used in systems such as beam instrumentation, cryogenics and power converters.

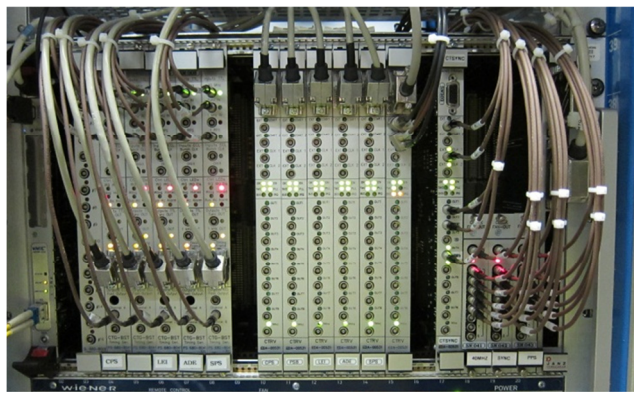


Figure 1: Timing crate controlling LHC.

Only one in four of the currently used module types are commercially available, while the others have been purposely designed at CERN. Furthermore, one in four of the module types are considered obsolete, meaning that existing modules can be maintained, but they cannot be used for new installations. These modules may have been designed up to twenty years ago and are still stocked in limited quantities. In most cases they can be repaired, but cannot be ordered or be re-produced in-house as the electronics components are no longer available.

STANDARDS FOR NEW DESIGNS

To replace the obsolete modules and to progress with new control and timing technologies, there is a continuous flow of new designs to be made.

To ease maintenance it has been decided in 2009 to base new designs on two platforms: VME64x [1] and PICMG 1.3 [2]. The latter defines an industrial type of PC that uses the PCI Express bus for its plug-in cards.

Most of the functionality that is required by the equipment groups is needed for both platforms and therefore every new device, such as a Time-to-Digital Converter (TDC) or an Analog-to-Digital Converter (ADC), would need to be designed twice. Potentially for *n* bus types and *m* functions required, *n* times *m* designs are needed.

If it were possible to put the specific required functionality on a mezzanine card, then with a single design of a carrier card for each platform, the number of designs would be reduced to n + m. The carrier card will contain basic functionality that is needed for every design,

#Erik.van.der.Bij@cern.ch

such as the bus interface, an FPGA, memory and other support functions while the mezzanine would only contain the specific functionality required.

Fortunately in 2009, the FPGA Mezzanine Card (FMC) standard ANSI/VITA 57.1 [3] was just published by the same organization that is responsible for the VME specifications. This standard defines a mezzanine card of around 84 mm by 69 mm in size and uses one of two connector types that have either 160 or 400 pins. There is no protocol defined for the signals on the connector as they should connect directly to an FPGA on the carrier board. As an example, Figure 2 shows such a carrier board for FMC mezzanines and uses a Xilinx Spartan 6 as FPGA. The mezzanine cards are supposed to contain only I/O logic such as buffers or an ADC, while a small front-panel space is available for external connectivity.

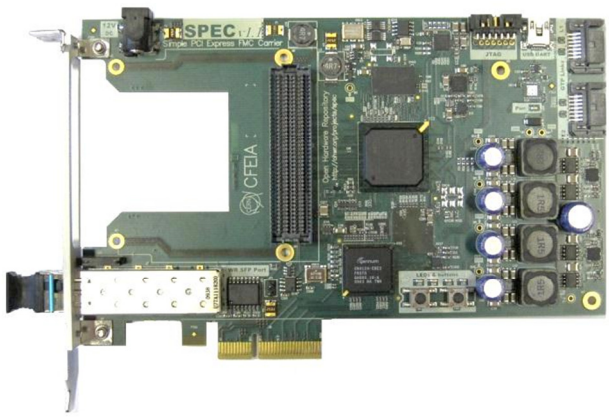


Figure 2: PCI Express carrier for FMC mezzanines developed as Open Hardware.

As many mezzanines and carriers use the same functions, the logic blocks inside the FPGA should be re-usable. To interconnect these blocks we decided to standardize on an internal bus called Wishbone [4]. Wishbone is a simple address/data bus meant to be used for system-on-chip designs and is an open standard. We collaborated in the update of the Wishbone B4 Specification to include a pipelined mode that enhances communication with high-latency high-throughput devices, such as DDR RAM controllers. Many ready-made building blocks are already available, such as Wishbone to I2C or to one-wire bus protocols.

Of course not all new designs for the control system use the FMC standard. E.g. level converters needing a large front-panel space for connectors are designed in the VME64x format.

NEED FOR OPEN HARDWARE

When new hardware is needed, in a few cases it could be bought directly from industry. This would be a resource gain as it is designed, built and tested already. At the same time it likely has proven its worth by many users in different applications. However, ready-made modules don't always have the exact functions we need. E.g., even for simple ADC modules CERN needs specific trigger modes or input capabilities that are not usual. Furthermore, when a bug is found, it may be hard to get a correction from the company as CERN's applications are very complex and the reported bugs may be difficult to reproduce. Ultimately, as there is normally no access to design documentation, it is impossible to help in solving problems.

It would be valuable if the advantages of both commercial and custom hardware could be combined. We believe we have found a way to do this by using the concept of "Open Hardware", similar to the concept of "Open Software" that has proven its worth.

The main ideas behind the Open Hardware paradigm are:

- All specifications and design files are published to benefit from peer review and to enable remote collaboration.
- All detailed production files for the hardware, including PCB production files, precise bill-of-material and assembly instructions are published so that anyone can reproduce the hardware.
- Peer reviews are actively sought for to improve the designs and make them better re-usable.

OPEN HARDWARE REPOSITORY

To allow sharing of design information, a simple web site or mailing list is not enough. Therefore, in 2008 a custom development of a collaborative tool was made and the first Open Hardware Repository (OHR) site went public in February 2009. One year later, the site was moved to the Open Source program called Redmine while in February 2011 the data was moved to a 'fork' of Redmine called ChiliProject [6]. The OHR website ohwr.org [7] now contains for each project modules such as a Wiki, a file repository, a news page and notably an issues list to keep track of found problems and required changes (see Figure 3). It is truly open as anyone can access all information.

One thing is to have the final design information available; another thing is to understand the reasoning behind certain design choices. Actually we try to have this information in the form of dedicated documents and by keeping an archive of the mail exchanges between the collaborating engineers.

At the end of August 2011, the OHR site hosted 45 active projects of which 37 are initiated by different CERN groups, while 8 are from other institutes or companies. On average each project mentions 3.6 persons as developer. About thirty projects describe hardware designs, often with associated software. Another twenty projects document re-usable IP core designs that will speed up many hardware projects. Also general software tools such as a production test environment and utilities to test the performance of ADCs are hosted.

On the OHR site there are around twenty projects related to FMC developments. E.g., one can find the complete designs of carrier boards in VME64x and PCIe format. As FMC mezzanines one can find projects of

Figure 3: Open Hardware repository project page of the Simple PCI Express FMC carrier.

ADCs using different sampling speeds (100 kSPS, 100 MSPS) and TDC and Fine delay modules with a resolution of 1 ns or better. These projects are initiated by CERN while a reconfigurable hardware interface for computing and radio (RHINO) is initiated by the University of Cape Town in South Africa. Bristol University in the UK started a Trigger/Timing Logic Unit in FMC format for use in a beam-telescope.

Examples of designs that do not use FMC are a small footprint single board ARM-based computer running Linux and a TTL to NIM level converter board in VME format. Also full designs of a network switch for the White Rabbit timing network can be found.

As many projects can re-use much of the functionality that is implemented inside the FPGAs, there is also a wide range of so-called IP-cores many of which interface to the Wishbone bus: examples are a DDR3 controller, Gennum GN4124 core, VME64x core, Wishbone serialiser and Wishbone slave register generator. Also an LM32 RISC processor core and a TDC that can be implemented inside an FPGA are fully documented in the OHR.

To the users the advantages of using the OHR are clear: not only does the collaboration tool allow one to easily work with known colleagues, but it also allows to find in a single place similar designs, examples of well documented projects and of course other engineers that have the same attitude of sharing and helping.

CERN OPEN HARDWARE LICENCE

Even for Open Hardware it is necessary to have a licence that defines the conditions under which a licensee will be able to use or modify the licenced material. Although many licences exist for developing Open Software (GNU, GPL, etc.), it appeared that there was none usable to cover hardware developments. For this reason the Knowledge Transfer Group developed the CERN Open Hardware Licence (CERN OHL) [8] that is compliant with the Open Source Hardware (OSHW) definition criteria [9]. In the spirit of knowledge sharing and dissemination, the CERN OHL governs the use, copying, modification and distribution of hardware design documentation, and the manufacture and distribution of products.

It shares the same principles as open-source software: anyone should be able to see the source (the design documentation in case of hardware), study it, modify it and share it. In addition, if modifications are made and distributed, it must be under the same licence conditions – this is the 'persistent' nature of the licence – which ensure that the whole community will continue benefiting from improvements, in the sense that everyone will in turn be able to make modifications to these improvements.

Although the CERN Open Hardware Licence was originally written for CERN designs hosted in the Open Hardware Repository, it is also used by other designers wishing to share design information.

EXPERIENCE WITH INDUSTRY

It was expected that the Open Hardware paradigm would change the model of how to work with industry. When commercial hardware was needed for CERN's accelerator control systems, it was usually obtained from companies with thirty or more staff. Now as the design information is open, there is no risk that a design will be unmaintainable when a company closes or an engineer leaves. This allows CERN to work with companies that are often much smaller in size. In the past two years, to speed up the development of specialised hardware and notably for the development of firmware of interfaces to Wishbone or VME64x, we have used the services of over twelve companies in eight different countries.

We have also seen that a large company is interested in developing hardware based on the White Rabbit timing protocol specification for which several Open Hardware projects are on-going. The company would not reveal its own independent design, but is willing to support the project in other ways such as standardisation of the protocol and reviewing the different designs. Another company will likely adapt the PCI Express FMC carrier to

use the PXIe bus and republish this design under the CERN OHL. This new card will be able to profit from all FMC mezzanine cards and the associated firmware and software developments too.

Finally, for the first tender process for CERN of the production and support of 70 PCIe FMC carrier cards, out of the seven companies requested, five did reply. These companies, all having already PCIe products in their catalogue, indicated that they would like to produce extra and sell them as a product of their own, exactly as was intended by the Open Hardware paradigm and licence. Actually any company, even if it did not win the contract, may produce the designs, stimulating a competitive environment. As it is likely that the FMC mezzanine cards will be produced by different companies than those producing the carriers, the Open Hardware concept may promote cooperation between companies.

One of the first designs in the Open Hardware repository is the nanoFIP, an interface chip for the WorldFIP industrial fieldbus. It is likely that this FPGA, that replaces an obsolete commercial device, will be used by a company to renovate a WorldFIP installation inside a train. Also for this company the full availability of the design information is crucial.

FUTURE WORK

As more and more projects are using the OHR, it will be important for engineers to be able to easily find designs for re-use. Currently a simple text based search is possible, but this may not be enough once hundreds of designs are available. The quality of the projects in the OHR will also be important. Already now some projects are very complete and active while other projects have only little information available. Currently this is covered by a status field that shows if a project is in planning state, mature, in alpha, beta or released. This may not be enough.

To allow sharing of design information such as schematics and the PCB layout, it should be possible for everyone to use the same tools that were used to generate this information. We believe that there are currently no free, open tools that are usable enough to make very complex designs and therefore we are looking into possibilities to raise one of the existing free tools to the required level, or to develop another one. Similarly we are helping to add VHDL support to the Icarus Verilog simulation and synthesis tool. Of course these developments will be made with the Open Software paradigm and we hope to attract other interested engineers to help in doing these developments.

CONCLUSIONS

The BE-CO Group at CERN is responsible for electronics that needs to be supported for tens of years. In some cases commercial hardware can be used but has the disadvantage of being black boxes that can only be supported by a single company.

By developing Open Hardware the designs function exactly how we need and in principle anyone can make improvements. Outside specialists may help and the peer review gives the potential for having really good designs. Also one is not tied to a single company for the production and support.

The CERN Open Hardware Licence paved the way for a solid legal ground, while the developed ohwr.org site allows engineers from different sites to easily collaborate on designs and to make the information public.

Having started with designs based on well-established standards such as VME64x, PCIe, FMC and the FPGA-internal Wishbone bus has helped to attract users and to make modules that are easily re-usable.

Finally, it has been shown that small engineering businesses find good opportunities to participate in the design efforts, while companies of various sizes have shown interest in producing and supporting Open Hardware designs in the same way as their own products.

Almost three years of experience show that the OHR environment and the ideas of Open Hardware are stimulating to engineers as it allows the sharing of their knowledge and results in better quality and better documented designs. For companies it gives new opportunities that may need to change their way of operating.

Designed using the Open Hardware concept, the PCIe carrier and 100 MSPS ADC mezzanine are being produced by industry, ready to be installed in CERN's control systems in early 2012. The nanoFIP fieldbus interface chip is also finished and is being designed into control systems. The public nature of this design attracted a European company to re-use it. Other modules, such as the VME64x carrier and 100 kSPS ADC, fine delay and TDC mezzanines are available as working prototypes and will be produced and supported by industry. This base set of modules, together with the supporting IP modules and drivers that have been made will allow rapid development of other modules that will be required by CERN's control systems and those of other installations.

REFERENCES

[1] ANSI/VITA 1.1-1997, VME64x Extensions
[2] PICMG 1.3, System Host Board PCI Express
[3] ANSI/VITA 57.1-2008 (R2010), FPGA Mezzanine Card (FMC) Standard
[4] WISHBONE, Revision B.4 Specification, OpenCores, 2010, http://opencores.org/opencores,wishbone
[5] P.Alvarez et al., FPGA mezzanine cards for CERN's accelerator control system, ICALEPCS 2009
[6] ChiliProject, https://www.chiliproject.org/
[7] Open Hardware Repository, http://ohwr.org/
[8] CERN Open Hardware Licence, CERN, 2011, http://ohwr.org/cernohl
[9] http://freedomdefined.org/OSHW

CHALLENGES FOR EMERGING NEW ELECTRONICS STANDARDS FOR PHYSICS*

Raymond S. Larsen, SLAC, Menlo Park, CA 94028, U.S.A.

Abstract

A unique effort is underway between industry and the international physics community to extend the Telecom industry's Advanced Telecommunications Computing Architecture (ATCA and MicroTCA) to meet future needs of the physics machine controls, instrumentation and detector communities. New standard extensions for physics are described which have been designed to deliver unprecedented performance and high subsystem availability for accelerator controls, instrumentation and data acquisition. Key technical features include an out-of-band imbedded standard Intelligent Platform Management Interface (IPMI) system to manage hot-swap module replacement and hardware-software failover. New software standards or guidelines are in development which will extend the reach of platform independent software standards to simplify design of low level drivers. Efforts to make the new standards broadly available in the marketplace through lab-industry collaboration are discussed.

INTRODUCTION

Several hurdles face the development and market acceptance of any new standard. In the case of earlier distinctly lab standards like NIM, CAMAC and FASTBUS the physics market itself banded together a priori to express a need and then develop an in-house standard. Success was guaranteed mainly by a simultaneous high rate of growth of both new labs and new front end and computing chip technologies. These standards were not universally adopted but had a reasonably high acceptance rate over several decades. As time went on the VME standard took over in the labs, supported by a strong military market focus. In 1994 ATCA and MicroTCA telecom standards were released and several labs working on next-generation machines became convinced that it was time for extension of this high availability high performance platform into physics controls, instruments and data acquisition. In early 2008 DESY made a decision to explore the platform for the recently approved X-Ray Free Electron Laser (XFEL). At the same time the International Linear Collider (ILC) R&D team chose ATCA as its technical and cost model. In early 2009 several labs joined the PCI Industrial Computer Manufacturer's Group (PICMG), an open-source standards organization of 250 companies, to develop formal standard extensions for physics. Goals included increased I/O channels and board size for analogue functions; rear transition modules with managed interfaces to enable hot-swap operation; and precision machine time synchronization and event triggering architectures. At the same time a software working group was launched to work on standardizing application interfaces, architectures and protocols and generic low level drivers for common modules such as ADCs; and to explore industrial high availability auto-failover software techniques for N+1 redundant systems in which a computer node has a backup processor and hub switch for example, or a backup timing network and generator-receiver at the node level.

The two major hardware standards have passed PICMG approval and are in publication, while software efforts continue with a dedicated core group. Exploration for controls is growing at several laboratories led by DESY, SLAC, IHEP and ITER. Several detector groups are exploring xTCA for LHC upgrades and several non-high energy physics machines including XFEL at DESY, LCLS at SLAC and others.

The major challenge today is to mature the new standard sufficiently quickly to meet needs of emerging new machine opportunities, particularly programs such as LCLSII at SLAC, the European Spallation Source in Europe and a number of smaller machines. The remainder of this paper will review progress to date and show examples of lab-industry collaboration to provide core infrastructure and common applications support.

TECHNICAL JUSTIFICATION FOR XTCA FOR PHYSICS

The initial attraction of xTCA for the 20-mile linear collider was based on bandwidth and throughput to cover machine and detector applications for the next 1-2 decades; and on its dual redundant Gbps serial backplane to achieve 0.99999 availability at the crate (shelf) level. The large ATCA card was seen to be most suited to special high throughput processor engines for fast RF feedback and detectors, while smaller more economical MicroTCA platform seemed best suited to Control systems supporting a variety of low and high speed applications in the same crate, e.g. medium speed RF feedback and slow DC magnet or vacuum controls and monitoring. A serially-networked xTCA architecture also easily accommodates so-called *network-attached devices* such as beam monitoring instruments distributed along a large machine at low density communicating only by serial link. A third attraction is the ability to extend IPMI management functions into non-xTCA devices, such as pulsed modulators and magnet power supplies, to bring unprecedented diagnostic information into control rooms and maintenance service centers.

*Work supported by US Department of Energy Contract DE-AC03-76SF00515

In summary, xTCA offers the following features:

- ATCA large card platform ideally matched to high bandwidth and throughput DAQ applications
- ATCA accommodates up to eight AMC mezzanine cards
- AMC mezzanine cards adapt easily to MicroTCA platform with similar bandwidth performance if desired
- MicroTCA real estate is cost effective for both very high speed applications as well as a high density of slow speed applications typical of industrial control
- Standards maintained and advanced by large industry consortium
- Consortium now includes laboratory members to advance features for both physics and non-physics applications
- Applications in non-physics areas including telecom potentially increases volume, improves market stability and decreases unit cost

NEW INFRASTRUCTURE STANDARDS FOR PHYSICS

Technical Features

The first standard extension, PICMG3.8, provides a standard hardware IO and management interface to the large ATCA card, which was left unspecified by PICMG. PICMG since has added an IPMI interface but no standard connectors. A new power and management connector was developed by the physics group for the standard. The interface is expected to cover a range of perhaps 80% of all applications. Where justified, the company or physics designer is permitted to choose a different connector which however compromises the physics goal of *interoperability* of designs made by different labs and companies.

The first test application at SLAC of PICMG3.8 is shown in Fig. 1.

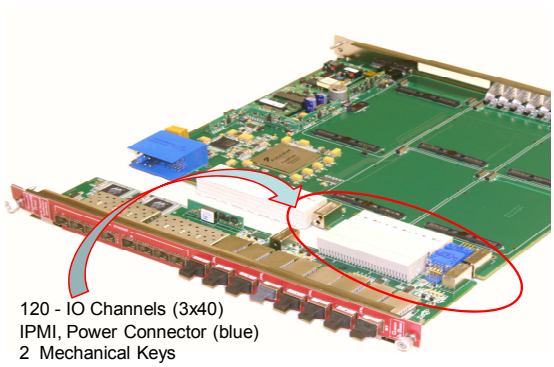


Figure 1: New ATCA RTM Interface PICMG3.8.

The second standard extension on MicroTCA specifies an entirely new 12-slot double-wide crate (shelf) and RTM with power and management controlled from the host AMC. The RTM card is identical in size to the AMC card but flipped vertically to properly align the rear panel with the in-line rear card guides. The IO interface is two 3-row 10 column channels (shielded 100Ω pairs) of which ten pairs are used for power, IPMI and JTAG connections. The concept is shown in Fig. 2 and industry hardware in Figs. 3-5.

Only the AMC connects to the backplane containing all the standard dual star interconnections plus a new layer for triggering and interlock sums. In addition a group of *extended options area* standard lines are specified for point-to-point precision clocks or triggers. The dual star redundant backplane is shown in Fig. 6.

The RTM connects to the host AMC only through its IO connector. Both the AMC and RTM are hot swappable although to swap the RTM all cables must first be disconnected. The RTM is expected to need replacement much less often than the more complex AMC.

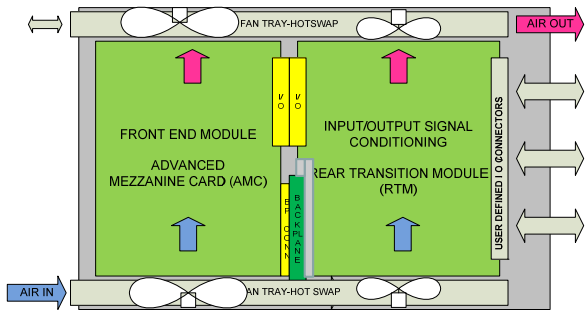


Figure 2: AMC-RTM-Shelf Concept MTCA.4.

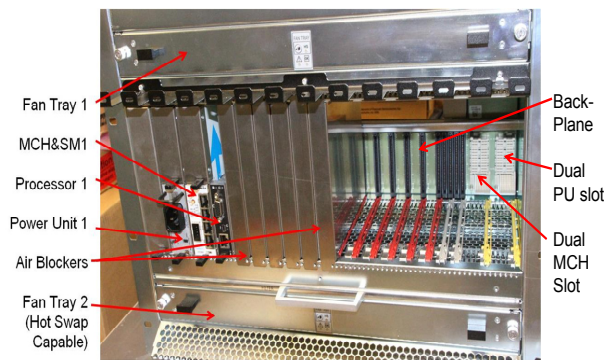


Figure 3: 12-Slot Redundant Shelf MTCA.4.

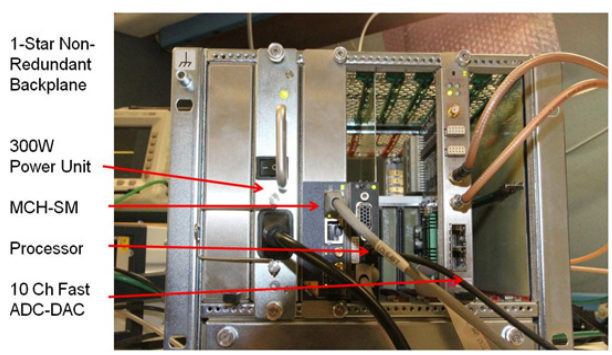


Figure 4: 6-Slot Development Platform MTCA.4.

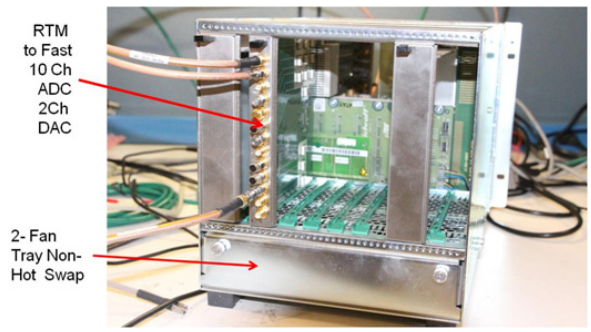


Figure 5: Rear of 6-Slot Shelf with Fast Digitizer RTM.

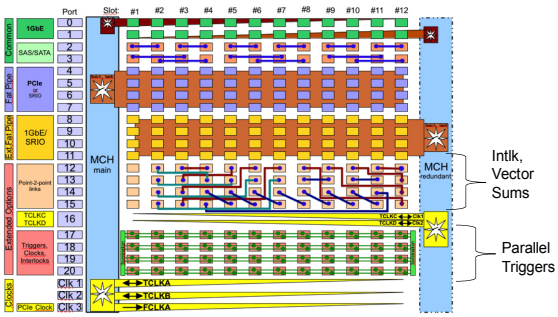


Figure 6: 12 Slot Redundant Backplane MTCA.4.

Progress Toward Infrastructure COTS Availability

One of the important metrics for the viability of any standard unless wholly developed and supported internally is availability of Commercial Off The Shelf products and servicing. Table 1 summarizes progress in the number of vendors supporting key components of the new standard, which officially is just being made public for the first time. Clearly the numbers are small but although no names are mentioned a significant number of major ATCA vendors contributed to the development and are fully engaged in product development.

Table 1: COTS Infrastructure Progress MCTA.4

Infrastructure	Description	COTS Availability
Development Shelf non-redundant	6-payload shelf with PU, integral cooling fans	2 vendors
Station Node Shelf dual star redundant	12-payload slot shelf , hot-swappable fan tray(s)	3 vendors
Modular Power Supplies	12V Power Units (PUs) 300/600/900W	2-4 vendors
Hub Controller (MCH) – full featured for timing needed	MCH Controller w/ integral IPMI shelf manager, hot-swap, access to radial timing option	2 vendors Switches for radial timing lines need development
IO Controller Processor (IOC)	Generic AMC processor running Linux, EPICS	2+ vendors
Timing Module	1 or 2-wide AMC (SLAC needs EVR compatible, needs adaptation)	1st units available (U. Stockholm), need COTS sources (2)

Note that the last item on the list is a timing module for MCTA.4, which has been developed in a collaboration of DESY with University of Stockholm but only recently is completed to where commercialization can be considered. Two versions are in progress, a single wide Tx-Rx unit suitable also for use on an ATCA Carrier board, and a double-wide for MTCA.4 that will allow additional channels to be distributed outside the shelf via the RTM. Small quantities of the first have been delivered and are operational, while the design of the second is in progress for completion in late 2011. Commercial sources are interested and working toward solutions also. The single wide unit and double-wide concept are shown in Figure 5.

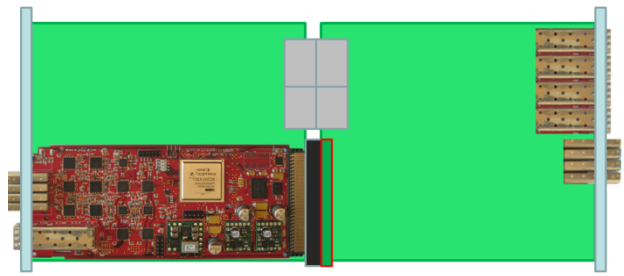


Figure 7: MTCA.4 Tx-Rx Timing Module Prototype (Stockholm University & DESY) and Double Wide Concept (in design, Stockholm).

LAB-INDUSTRY APPLICATIONS DEVELOPMENTS ON MTCA.4

The next important metric is the commercial availability of key generic AMC modules:

- High performance fast digitizers for DAQ and LLRF applications
- FPGA modules to process instrumentation tasks such as interlocks, beam position, beam charge etc.
- AMC Adapter modules for common industry standards e.g. PMC, Industry Pack, FMC etc,

With these three generic modules and customized RTMs developed by either labs or by industry, the bulk of controls and monitoring requirements for an accelerator can be instrumented. Some AMC adapter applications include stepping motor drivers, temperature, beam loss, wire scanners, fast frame grabbers, etc. Examples are shown in Figures 6-13.

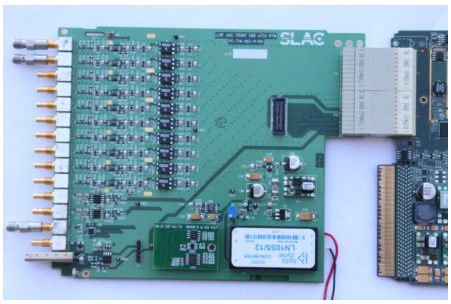


Figure 8: 25 MHz IF RTM (SLAC) for 10 Ch 16 bit 125 MSps COTS Digitizer (Struck).

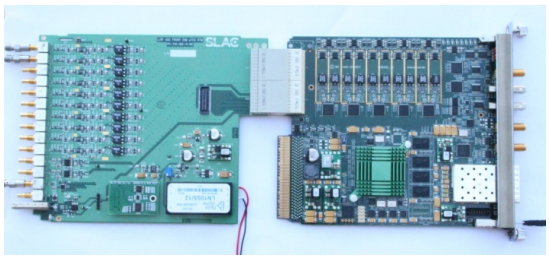


Figure 9: RTM passes signals at 25 MHz from 2.856 GHz down-converter in external RF Front End chassis.

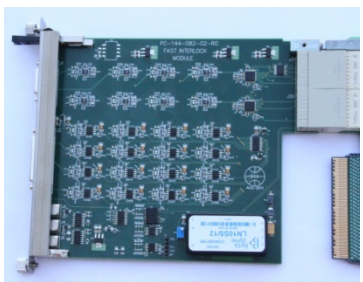


Figure 10: Fast/Slow ADC RTM (SLAC) mates to COTS Generic FPGA AMC (TEWS 651) for Klystron Interlocks Processing.

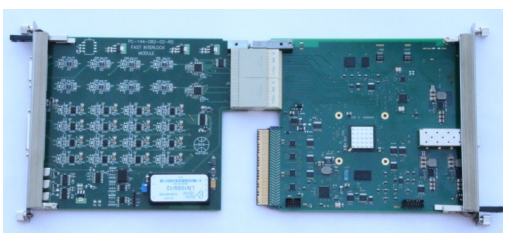


Figure 11: Mated Pair handles 8 Channels 60 MSps and 16 Channels 2 KSps Interlock Input Channels.

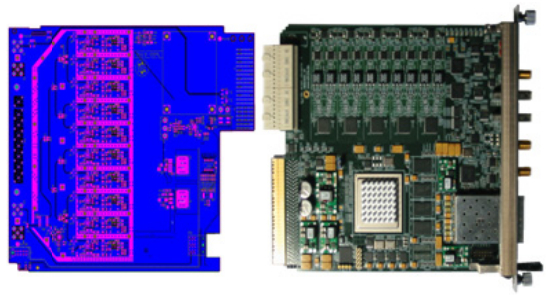


Figure 12: IF Down-Converter RTM to Struck ADC-DAC (DESY).

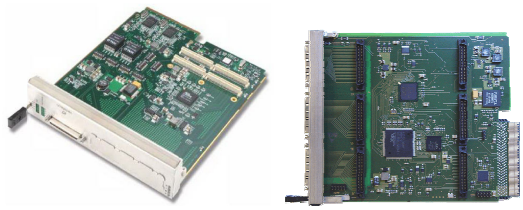


Figure 13: PMC Double Wide AMC Adapter (Vadatech) and 3-IP AMC Adapter (TEWS)*
*Now available on MTCA.4 version with RTM.

Progress Toward Generic Applications COS Availability

Table 2 summarizes progress toward development of COTS suppliers for generic AMC modules. There are 1-2 vendors in existence or in development for the items shown. It is the goal of the PICMG program to develop at least two commercial suppliers for critical products so the Physics Technical Committee has chosen to apply this to both basic infrastructure and basic generic products.

The right hand column is a partial list of the instrumentation applications that can be served by this limited list of generic products. In other words, a small number of complex AMC products is highly leveraged to a number of applications which significantly reduces development costs and speeds time to project-readiness (or to market for vendors).

RISK FACTORS AND RISK REDUCTION

General Observations

Most if not all new accelerator projects, by the time they have spent years in development and battling for approvals, are highly averse to potential time and cost risks associated with a new standard. The decision by a single large project or a number of projects to adopt a new controls and instrumentation platform, an event that occurs approximately every two Sun-Spot Cycles in physics and industry, must be solidly based on enough prior work to demonstrate the technology, especially the

Table 2: COTS Generic Applications Progress

Generic AMC	COTS Availability	RTM Adapters
10/2 Ch ADC/DAC 16 bit 125 MSPS	1 vendor available 2nd vendor due end FY11	• RF-IF down-mixers • BPM adapter • Photodiodes
4 Ch ADC AMC 14-16 Bit 125 -500 MSPS	1-2 vendors in development	• BPM single bunch • BPM multi-bunch • Beam intensity Toroid • Beam Length
FPGA Virtex/Spartan, FMC optional	1 vendor in development	• Interlocks ADCs12 bits, 8 ch @60 MSps, 16ch@2KSps • Wire scanner interface
AMC Industry Pack Adapter (2-3 IPs)	2 vendors in development	• Stepping motor control • Vacuum control-monitoring • Temp control-monitoring
AMC PMC Adapters	2 vendors	• Timing Rx adapter • Frame grabber adapter

software, and the hardware-software costs by achieving basic COTS availability.

ATCA is actually a decade old and it is in mid-ramp in Telecom which now is claiming in excess of $1B per year of product in a $10B per year market. A small part of this is in MicroTCA which developed as a standalone in a second phase of development so is much less mature. Therefore one has to admit that establishing any new platform takes time and investment and involves risk.

On the other hand, technologies advance rapidly, obsolescence is inevitable, and at some point all platforms have to be replaced. Laboratories must plan the inevitable changes to minimize risks to ongoing and emerging products when they do occur. Otherwise, as many laboratories have learned, a failure to modernize adiabatically during the long life of huge accelerators can cause insurmountable barriers of time and cost when they become truly unavoidable.

Lab Industry Collaborations a Tool for Risk Reduction

New platform development is a large costly undertaking and requires a dedicated community of champions to achieve. The PIMCG consortium is such a community which has had large success for many years. The xTCA for Physics standards efforts are only two years old but have already saved large development time and cost by the collaboration. Coordinated joint development of basic industry support for both infrastructure and generic applications is already paying dividends. There is major work ahead for any lab adopting the standards, but a great deal is already done and being made available to the entire community.

NEXT STEPS: SOFTWARE EXTENSIONS

The xTCA for Physics collaboration has achieved its initial goals and now must bring the same enthusiasm to developing streamlined software modules, tools and supports. The major goal is to standardize modules to address imbedded FPGA based systems and to simplify development of new applications by such standards or best practices guidelines. The objective as in hardware is to make products built in both labs and industry more compatible and *interoperable* as has been achieved through the IPMI management system for the hardware platforms.

CONCLUSION

The xTCA for Physics Lab-Industry collaboration has achieved the launching of a new hardware and software architecture for physics. Its ultimate success depends on many factors and will vary across many applications, but it is now available as new system that is already being explored and adopted at several labs. The next major effort must be to extend software compatibility at low levels of hardware, which has never been achieved satisfactorily in any present standard.

The same approaches that underlie the xTCA architectures are also extendable to non-xTCA devices such as Megawatt class power supplies and pulsed modulators for RF systems, AC and DC high availability systems etc. A judicious use of extra dual or N+1 redundancy can enable us to visualize in future "machines that never break." Of course they will break, but the high availability architectures of xTCA applied broadly (which linacs have inherently) can make render breakages of little or no impact in keeping the machine running.

Similarly Intelligent Platform Management diagnostic extensions beyond basic control and instrument modules into all major power systems and remote instruments will have a major impact on machine uptime and reduce the time, effort and number of people required for servicing.

ACKNOWLEDGMENTS

This paper represents the work of an entire community far too many to name.

The PICMG xTCA for Physics collaboration was sponsored by DESY, IHEP, FNAL, SLAC and now has added lab members CERN, IPFN Lisbon, ITER, KEK, LBNL and Sincrotrone Trieste. Cypress Point Research and TripleRing Technologies were industry sponsors.

Enormous credit is due to key contributors from both labs and industry, including Schroff, ELMA, Positronix, TE Systems, Pentair-Schroff, Performance Technologies and others from among over 40 industry members

The DESY XFEL teams headed by Kay Rehlich and Stefan Simrock (now with ITER) made ground-breaking technical contributions in ATCA and MTCA.4 early applications and design.

Special thanks to Robert W. Downing, the driving force with Dick Somes who constructed and edited both specifications; to Stefan Simrock and A.P. Gus Lowell who lead the Software team; and to Joe Pavlat, PICMG President and Doug Sandy, PICMG Technical Officer for unwavering support. Sincere thanks to the xTCA pioneers at SLAC: Z.Geng, A. Young, D. Brown, C. Yee, C. Xu, D. van Winkle and E. Williams.

UPGRADING THE FERMILAB FIRE AND SECURITY REPORTING SYSTEM

C. King, R. Neswold
FNAL†, Batavia, IL 60510, U.S.A.

Abstract

Fermilab's homegrown fire and security system (known as FIRUS) is highly reliable and has been used nearly thirty years. The system has gone through some minor upgrades, however, none of those changes made significant, visible changes. In this paper, we present a major overhaul to the system that is halfway complete. We discuss the use of Apple's OS X for the new GUI, upgrading the servers to use the Erlang programming language and allowing limited access for iOS and Android-based mobile devices.

INTRODUCTION

FIRUS is an acronym for **F**ire **I**ncident **R**eporting and **U**tility **S**ystem. It was developed at Fermilab in the early eighties using the technology available at the time resulting in consoles that run on MS-DOS based PCs using Digital Research's GEM graphical user interface and servers running Motorola's VersaDOS. Consoles communicate with servers via ARCNET and are segregated from Fermilab's main network for security reasons. This has resulted in a reliable, high-availability system that is still in use today.

OVERVIEW

FIRUS is comprised of front-ends, minis and consoles which all communicate via a private ARCNET network. The network is divided into eight trunks as shown in Figure 1. The front-ends are connected to all eight trunks, while the minis and consoles are only connected to one.

Front-Ends

At the heart of FIRUS are the two front-ends, "blue" and "red". The blue front-end is considered the primary and the red front-end is the backup. Under normal conditions, both blue and red divide the load of scanning minis and communicating with consoles equally. They each hold a full copy of the device database, which contains hardware addresses and alarm limits. Edits to the database go to blue, which then forwards the changes to red's copy.

FIRUS is designed to be fault-tolerant against a single front-end failure. The two systems monitor each other and, if one determines the other is unresponsive, it will assume responsibilities for all minis and consoles until the other front-end returns to service. This feature also allows us to take one system down for maintenance without impacting Fermilab's security and fire departments.

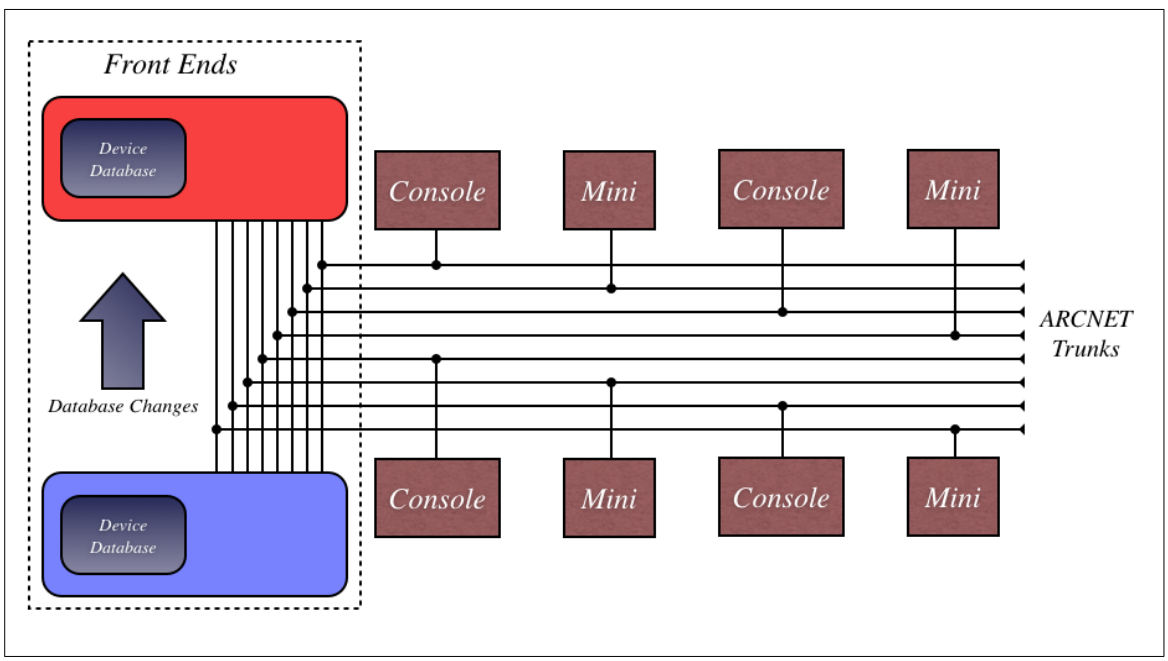


Figure 1: FIRUS Topology.

†Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

Minis

"Minis" are small, embedded systems distributed site-wide. There are 150 minis currently active in FIRUS. Each mini is controlled by an 80186 embedded processor which uses an ARCNET interface to communicate with the front-end systems. A system can contain up to seven daughter cards each of which provides analog or digital inputs. The analog cards provide 16 channels of 16-bit A/D conversion while the digital cards supply 16 bits of input. Each digital input is instrumented to detect shorts, open circuits and grounded signals. Minis are embedded software modules that connect to device hardware.

Consoles

All user interaction with FIRUS happens on the console. The important features of a FIRUS console are as follows:

- Alarm acknowledge and display
- Alarm logging
- Device database management
- Real-time parameter page display
- Data logging at multiple rates
- Real-time and logger plotting
- Synoptic picture displays
- Fully configurable
- Password protection for sensitive items

UPGRADING THE CONSOLE

The FIRUS console hardware hasn't had a refresh since it's original inception in the mid eighties. We are now reaching the point of obsolescence, thus creating a nightmare maintenance scenario. GEM compatible PC hardware was becoming increasingly hard to come by. GEM requires EGA which is fairly low resolution by today's standards and only supports a limited sixteen colours. Contrast that with the ultra high resolution, million colour displays of today and it's not hard to see that change is needed.

The combined memory issues with MS-DOS and GEM was also a source of frustration. Adding features and making changes were almost impossible. Eventually the FIRUS application had to be split into two separate executables because of GEM's 64K maximum resource data segment size, resulting in a stripped down FIRUS application and a utility FIRUS application known as UFIRUS. Because MS-DOS is not a multitasking environment, this split basically removed some features from FIRUS as a whole since most users would never run UFIRUS.

Mac OS X

As it became more of a challenge to obtain replacement PC hardware, more time was invested in determining a new platform for the FIRUS console. Windows XP, Linux with QT, Java and OS X were all possible, but OS X won out because of it's modern graphical interface, it's Unix core and it's powerful set of **free** development tools[1].

OS X has the benefit of being a direct descendant of NeXT Inc.'s OPENSTEP operating system[2]. It inherited from OPENSTEP the feature-rich, object-oriented interface as well as the beginnings of a build environment that Apple has masterfully crafted into what is now known as Xcode. Xcode presented a learning curve but proved to be well worth the effort. It was clear that not only was Apple catering to the end user, they were also making OS X attractive to the developer.

Design Goals

FIRUS has been in use by Fermilab since the early eighties, hence our most important design goal was to minimize the user's learning curve. User acceptance was very integral to the success of the upgrade. Most people are annoyed and/or frightened by change, especially with changes without fundamental improvement to the workflow. Obviously changing the graphical user interface poses some changes that may require the users some time to adjust, but overall we strove to minimize most changes unless deemed value added.

Other design goals included:

- No changes to front-end code so both GEM and new consoles could run side by side for verification purposes
- FIRUS will be one cohesive application (GEM was two separate applications)
- Add a kiosk mode to keep unprivileged users from switching away or quitting the FIRUS application
- Automatic software update distribution with versioning

ARCNET Challenge

After deciding on Mac OS X, iMacs and Mac minis were chosen as the hardware we would use to replace the current consoles. Both the iMac and Mac mini are the least expensive of the Macintosh line, yet were powerful enough for our needs. They also both have a very small footprint and are able to physically fit where any GEM console currently resides.

One of the more challenging aspects of the FIRUS console upgrade was how to get the desired Macintosh computers connected to ARCNET. Since iMacs and Mac minis were the chosen hardware, ARCNET cards were not an option. The search was on for a device that would bridge the gap. Our search turned up such a device. The USB22-CXB from Contemporary CONTROLS is an external USB to ARCNET bridge that is powered by the USB port thus eliminating the need for separate power, and it connected to directly to our coaxial-based ARCNET topology.

Alas, our excitement was short lived when we discovered that the USB22 was virtually unsupported on OS X. Contemporary CONTROLS had no plans to add support, so if we wanted to use the device we would have

to do it ourselves. Our first thought was to make the USB22 a true network device in order take advantage of built in network configuration in OS X. While this approach worked well (we were able to copy large files, stream video and browse web pages over ARCNET), it was not compatible with the FIRUS minis. The minis were unable to ignore the multicasted TCP/IP traffic and would become unresponsive. Since updating the software in the FIRUS minis was out of the scope of the upgrade, we had to abandon the network driver for a more FIRUS-specific solution. The final decision was made to use OS X's application level USB driver interface. Code for supporting the USB22 is built in to the FIRUS consoles, and as a result, the application can notify the user of network or device problems in a more timely manner.

Development Notes

Overall, the development of the new FIRUS console was an exciting project to work on. The Xcode integrated environment make easy work of designing the user interface elements. The source code editor aided code production by providing context sensitive help and autocompletion. The debugger worked flawlessly, and the instrument tools for memory issues assured that the high -availability goal for the console could be attained.

Xcode, however, wasn't our only learning curve. We also received lessons on usability. Just because certain standard features of OS X are indispensable in some applications, doesn't mean they will automatically be well received in the FIRUS console. One specific incident pertained to the basic alarm screen. We assumed that giving the users ad-hoc sorting by any displayed column would be a feature they couldn't live without. As it turned out, that was one feature they couldn't live with. FIRUS operators were so used to alarms lists that are sorted by decreasing time that they didn't want to even have the ability to change it and possibly misread the alarm screen. From that point on, we became more FIRUS user- centric in our user interface design.

FUTURE DEVELOPMENT

Now that the FIRUS console upgrade is complete and in the process of certification, we look toward future development for FIRUS. During development of the FIRUS console upgrade we discovered a number of front-end deficiencies as well as inaccurate behaviour.

Front-End Upgrades

In the lifetime of the FIRUS system, the front-ends went through one major upgrade. Right before Y2K, we upgraded the hardware from a 68K-based VME board to a multi-gigahertz Intel-based PC. The FIRUS software was ported -- not rewritten -- from Motorola Unix to a BSD Unix. The upgrade gave us a lot of CPU horsepower and much more memory to use than the original system. None of these extra resources have been exploited, yet. Now that the consoles have been updated, we look to modernize the front-ends again.

The software in the front-end was written in a time of limited CPU resources and memory constraints. The operating systems didn't support threads and the FIRUS processes weren't written to fork parallel processes to handle multiple requests. We feel that, with the resources available on the new consoles, a greater number of requests will be placed on the front-ends. They need to scale gracefully with the extra load, which the single threaded design won't be able to do.

Another shortcoming is that the tasks on the front-end don't communicate with each other when some global, system state has changed. The database task, for instance, doesn't inform the alarm task that an entry has changed. Instead, a console will see the change when the alarm task eventually rereads the database. It would be nice if state changes, like this, are synchronized better within the front-end.

Our decision is to rewrite the front-end software using the soft real-time, functional programming language, Erlang. Erlang was developed by Ericsson to use in their telephony equipment[3]. The language features concurrency primitives as part of the language and uses very light-weight processes to break a problem into simple pieces. Processes communicate with each other using message queues. The Erlang runtime is very rich and includes an ACID-compliant, distributed database (which will get leveraged in the upgrade.) There is also a web server module so the front-ends could generate and deliver web pages displaying status.

Remote Access/Mobile Devices

In today's well-connected world, WIFI and cellular data access are the norm. People have come to expect universal access to their data. Now the question arises. How do we securely deliver remote access to FIRUS.

We are currently testing a web-based remote alarm display that gives users access both on and off site. This is done securely by routing all traffic though a proxy server using a secure socket layer (SSL). All on-site access is automatically allowed. Off-site access is allowed only after password verification. This is currently in use by our on-site fire technicians. When testing contact circuitry, they are able to view alarm status on the web browser running on their portable scanning device. This has been very successful by reducing the testing time in the field.

We believe the future of the FIRUS console could include application development on mobile devices. Today's mobile devices are essentially ultra portable computers capable of running complex applications. With proper security considerations, we feel that many of the console features could be incorporated into applications for the iPhone and iPad and possibly some Android-based devices.

CONCLUSION

FIRUS has been a reliable, high-availability system that has been in use since the mid eighties at Fermilab. But

over the last few years, FIRUS started showing signs of it ageing architecture. In this paper, we presented the highlights of the FIRUS console upgrade as well as detailed what remains to be done to take FIRUS into the future.

REFERENCES

[1] "iOS Dev Center – Apple Developer", 2011, Apple Inc. http://developer.apple.com/devcenter/ios/index.action

[2] "Erlang Programming Language", 2011, Ericsson AB. http://www.erlang.org/

[3] "Mac OS X – Wikipedia, the free encyclopedia", 2011, http://en.wikipedia.org/wiki/OS_X

SARAF CONTROL SYSTEM REBUILD

E. Reinfeld, I. Gertz, I. Eliyahu, Soreq NRC, Yavne ANL ISRAEL

Abstract

The Soreq Applied Research Accelerator Facility (SARAF) is a proton/deuteron RF superconducting linear accelerator, which was commissioned at Soreq NRC. SARAF will be a multi-user facility, whose main activities will be neutron physics and applications, radio-pharmaceuticals development and production, and basic nuclear physics research. The SARAF Accelerator Control System (ACS) was delivered while still in development phase. Various issues limit our capability to use it as a basis for future phases of the accelerator operation and need to be addressed. Recently two projects have been launched in order to streamline the system and prepare it for the future development of the accelerator. This article will describe the plans and goals of these projects, the preparations undertaken by the SARAF team, the design principles on which the control methodology will be based and the architecture which is planned to be implemented. The rebuilding process will take place in two consecutive projects. The first will revamp the network architecture and the second will involve the actual rebuilding of the control system applications, features and procedures.

INTRODUCTION

SARAF is currently under construction at Soreq NRC [1]. It will consist of a medium energy (up to 40 Mev) high current (up to 2mA, CW, upgradable to 4mA) RF superconducting LINAC of protons and deuterons. The main accelerator assembly was built and delivered to the SARAF facility in 2007 and two beam lines were constructed by the SARAF team and completed in 2010. The project phase I is planned to be completed during the next couple of months and phase II is planned to start soon after (figures 1 and 2).

The accelerator control system was delivered while in development state and consisted of a number of semi-independent control systems, each designed to control a specific component of the accelerator [3]. Development ceased after delivering the system to SARAF team and was held in breakdown maintenance mode until the first two beam lines were constructed.

This article will describe the accelerator control system rebuilding project which is planned to transfer the control system from development mode into operational mode, and prepare it for phase II of the SARAF project.

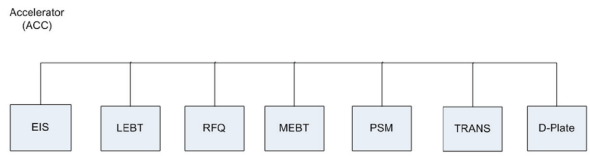


Figure 1: Accelerator overview.

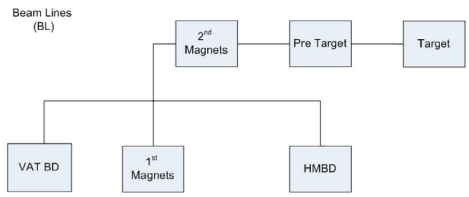


Figure 2: Beam lines overview.

PROBLEM DEFINITION

While preparations took place to the beam lines construction, the accelerator control system (ACS) was not developed further and was kept running with minimal effort and with no dedicated engineering staff.

As the beam lines construction advanced, it became apparent that the control system as a whole needs to be reviewed and assigned with control engineer team in order to "fix" the problems which were encountered by operators on a daily basis.

A review was conducted and as the system was further analyzed it became apparent that in order to prepare the facility for phase II of the project, the current control system which may have been fine for the development and commissioning stages must be overhauled and rebuilt.

The main issues which have to be solved can be summarized by the following list.

1. Lack of knowledge and documentation of the current control system. As the control system delivery was not completed there is close to no documentation at all on the various control components and mechanisms.
2. Fragile infrastructure. Key elements are missing from the control network infrastructure, which potentially can lead to severe consequences.
3. No methodology or strategy. Development of the control system was conducted "on the fly" and with no broad picture overview on how an accelerator control system should be built.
4. Waste of time and resources on keeping the system running. The operators have to deal with a control system which does not fully answer their needs and requirements.

In short, the system required a rebuilding effort that will stream line it into a solid operational control system that will provide a solid basis for further operation and development.

SOLUTION DEFINITION

To rebuild the system a methodical approach is planned, in order to address the issues described above systematically. Planning a strict method and following it in the implementation stages is important for standardization. An effort is made to complete each phase of the project as planned.

Revamping the infrastructure could be done in parallel, and was launched as a secondary project. This project is planned to upgrade the SARAF network capabilities in term of bandwidth, introduce network elements which were missing or not up to date, introduce cyber security protection suite tailored according to current standards for national infrastructure and should provide a solid foundation to the control system rebuilding project. The network revamp was defined as a necessary goal that is part of the control system base.

The control system rebuilding road map is described in figure 3.

Several constraints have to be taken into account when implementing the solution. The main constraint is that the rebuilding process will be done while the accelerator is still operational and a long period of shut down time is not planned. This means that the rebuilding process will be done on a living system and calls for careful planning and structuring of the rebuilding process, especially when migrating the software and tools, and when replacing specific applications.

IMPLEMENTATION

The project was divided into several consecutive phases. Each one consists of several action blocks which can be done in parallel to prevent bottle necks and to allow progress in various subjects by different teams.

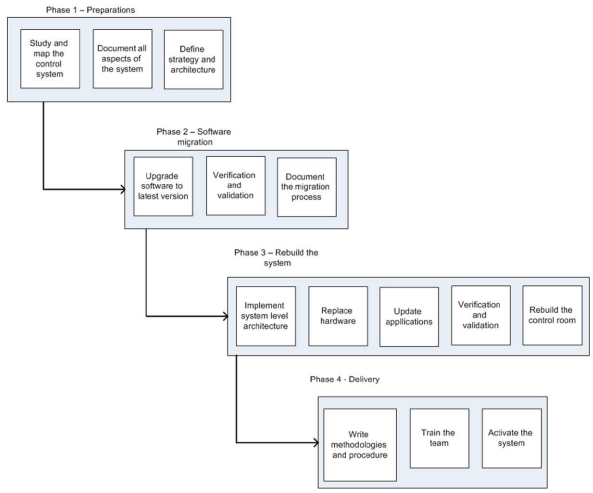


Figure 3: Project phases.

Phase A - Preparations

This phase creates a concrete control system knowledge base in the facility and defines the structure of the rebuilding project. This phase involves a thorough mapping and documentation of the entire control system. It involves reverse engineering efforts of the facility team members including the operators and engineers. At this point every major issue or bug found while reviewing the system will be cleaned and the applications will be updated wherever possible.

Mapping the system includes mapping I/O points, hardware, software and tools and reorganization of the system data.

Phase B – Software Migration

The SARAF control system is based on Labview 8.2.1 software. Some of the new control hardware is not compatible with this version as well as tool kits needed for new instalments. In this phase the software will be migrated to the latest stable version of Labview, and the entire application suite and control system will be migrated as well.

This will be a challenging step as the accelerator will be operational when the migration process takes place, and so both versions will coexist in the control system until all the applications are replaced.

Phase C – Rebuild the System

This phase will introduce new capabilities to the system, starting with a higher level of hierarchy to support flow control and implement the architecture designed in phase I of the project. This phase will see replacement of obsolete hardware, major update to all relevant application and should result in a stream lined and robust control system.

The last step of this phase will be the rebuilding of the control room according to modern design principals which will allow higher flexibility for operators and designers to work in.

Phase D – Delivery

The last phase will see the delivery of the control system to the SARAF team. After writing down operation methodologies, the team will be trained to use the system and the new tools introduced to it.

The following period will be to activate the control system and to monitor it for bugs and fine tune it according to operators needs.

SUMMARY

The SARAF facility has reached a maturity level to transfer it from the development state it was delivered, into an operational state. Two projects were launched to address the transfer, one to address infrastructure issues, and create a work environment which is stable and robust. The second is to rebuild the control system and create a basis for the future phases of the SARAF project, and to assimilate the engineering methodologies needed to operate and support a system of this kind.

REFERENCES

[1] http://www.soreq.gov.il/default_EN.asp.
[2] I. Mardor et al., "The SARAF CW 40 MeV Proton/Deuteron Accelerator", SRF09, September 2009, Berlin-Dresden, MOODAU04, (2009)
[3] I. Mardor et al., "The operation Concept of SARAF", LINAC'06, Knoxvill, August 2006, MOP033, p. 109 (2006).
[4] I. Gertz et al., "Status of the SARAF Control System", ICALEPCS2009, Kobe, October 2009, TUP109 (2009)

THE UPGRADE PROGRAMME FOR THE ESRF ACCELERATOR CONTROL SYSTEM

J. Meyer, JM. Chaize, F. Epaud, F. Poncet, JL. Pons, B. Regad, E. Taurel, B. Vedder, P. Verdier,
ESRF, Grenoble, France

Abstract

To reach the goals specified in the ESRF Upgrade Programme [1], for the new experiments to be built, the storage ring needs to be modified. The optics must be changed to allow up to seven meter long straight sections and canted undulator set-ups. Better beam stabilization and feedback systems are necessary for the nano-focus experiments planned. Also we are undergoing a renovation and modernization phase to increase the lifetime of the accelerator and its control system.

This paper resumes the major upgrade projects, such as the new beam position monitoring (BPM) system, the fast orbit feedback, the ultra-small vertical emittance and their implications on the control system. Ongoing modernization projects such as the solid state radio frequency amplifier or the high-order mode (HOM) damped cavities are described. Software upgrades of several sub-systems e.g. vacuum and insertion devices, which are planned either for this year or for the long shutdown period, beginning of 2012, are covered as well. The final goal is to move to a Tango [2] only control system.

THE ESRF UPGRADE PROGRAMME

In 2008, the Council of the ESRF launched the ESRF Upgrade Programme 2009-2018, an ambitious ten-year project. Funding for a first phase of the Upgrade (from 2009 to 2015) has been secured to deliver:

- Eight new beamlines, mainly with nano-focus, with capabilities unique in the world
- Refurbishment of many existing beamlines to maintain them at world-class level
- Continued world leadership for X-ray beam availability, stability and brilliance
- Major new developments in synchrotron radiation instrumentation

Producing nano-sized beams needs long beamlines, which at the ESRF will reach 120 metres. An extension of the experimental hall will be built to house the new beamlines.

Nano-focusing also requires a high reliability, stability and brilliance of the x-ray source. To reach the goals of the update program, the x-ray source also needs to be modified.

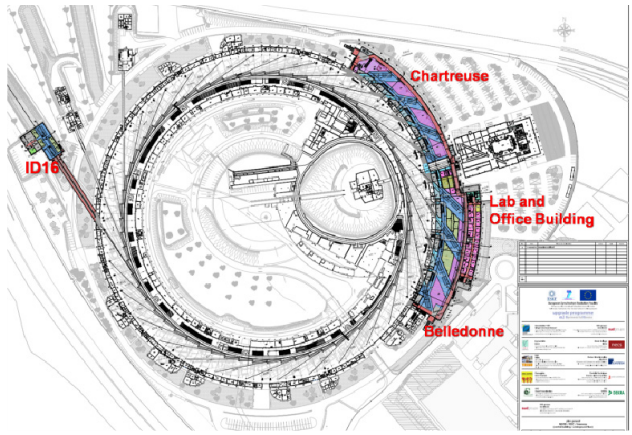


Figure 1: Experimental Hall Extension.

X-RAY SOURCE IMPROVEMENTS

Several improvements are already in preparation since 2009 in the framework of the ESRF Upgrade Programme.

BPMs and Fast Orbit Feedback

The old beam position monitoring system was already replaced at the end of 2009 with 224 Libera [3] measurement systems. With these high precision e-beam position measurements, the exchange of all 96 steerer power supplies and the installation of a fast, redundant communication network, we are now able to prepare a new fast orbit feedback system with a correction rate of 10 kHz. The result should be a better correction of e-beam movements.

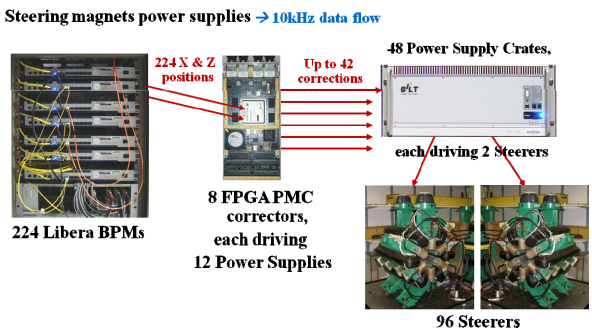


Figure 2: Fast Orbit Feedback System.

The feedback system is actually under test and should be operational at the end of 2011. The challenge for the control system was the progressive exchange of the steerer power supplies, during several months, without perturbing the operation.

Ultra-Small Vertical Emittance

A new algorithm for coupling correction was developed and implemented for the storage ring. Using this, together with the high precision Libera beam position monitors a vertical emittance of $\varepsilon_z = 4.4 \pm 0.7$ pm could be reached.

Two procedures to preserve the small vertical emittance during beam delivery were successfully tested: stable ε_z = 6-7 pm for a 7/8 +1 filling mode and 10 pm for a uniform filling.

During the last winter shutdown, 32 new skew quadrupole magnets have been installed on the storage ring to correct the coupling induced by insertion device movements. The goal is an ultra small vertical emittance of ε_z = 2 pm. The first already carried out tests are promising, and the new fast orbit feedback system will help this goal to be reached.

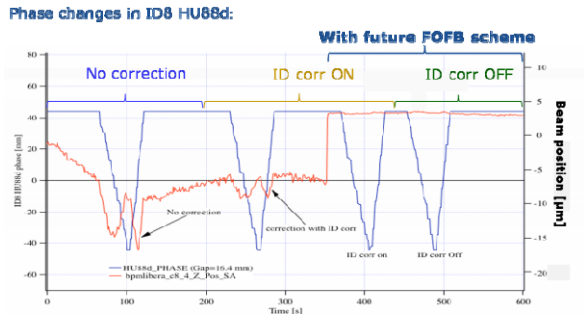


Figure 3: Correction of ID beam perturbation.

6m Straight Sections

The lattice of the ESRF storage ring was designed with 32 long straight sections of alternating high and low horizontal beta values. This currently provides 5 m of available space for insertion devices. The ESRF Upgrade Programme proposes to increase the length of selected insertion device straight sections from 5 to 6 or even 7 meters. This will provide a higher level of brilliance and increased insertion device flexibility. For example, longer insertion devices for a single beamline, sharing of the straight sections between two experimental stations using the canted undulator approach, reshuffling the RF cavities layout and freeing straight section space for new beamlines are possible uses of the increased straight section length.

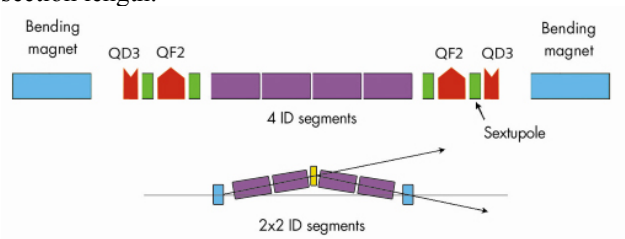


Figure 4: Uses of long straight sections.

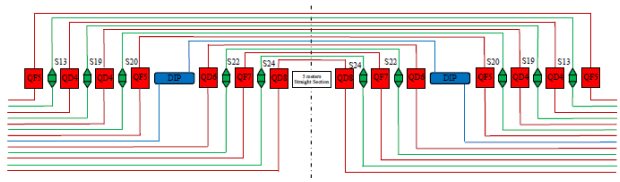


Figure 5: Initial, family-wise, power supply steering.

To allow such long straight sections, a lot of power supplies need to be changed from a family-wise control to individual control. The beam steering algorithmns had to be revised.

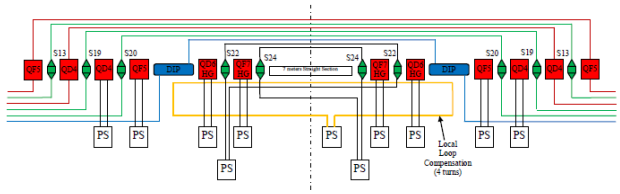


Figure 6: Power supply control for a 7m straight section.

High Power Solid State Radio Frequency Amplifiers and HOM Damped Cavities

The sub-system with the highest failure rate on the ESRF storage ring during the last years was the radio frequency system. To increase the long term reliability, two projects are in progress in the frame work of the Update Programme.

The replacement of the klystron based radio frequency transmitters with solid state radio frequency amplifiers and the development of new HOM-damped accelerating cavities.

Solid state amplifiers consume less power, have a high redundancy and need much less tuning effort compared to a klystron.

The first amplifier tower has been tested at the ESRF. Delivery and installation of the first full transmitter is planned for the end of 2011.

A careful software design is important, because the control system needs to combine the new and the old radio frequency controls in a coherent way.

Figure 7: Solid state RF amplifier at Soleil.

High order mode tuning of the accelerating cavities, especially at higher e-beam currents, is complicated to achieve. To overcome the limitation, a new design of a HOM-damped cavity [4] was made. The prototypes have been delivered to the ESRF and are currently under test.

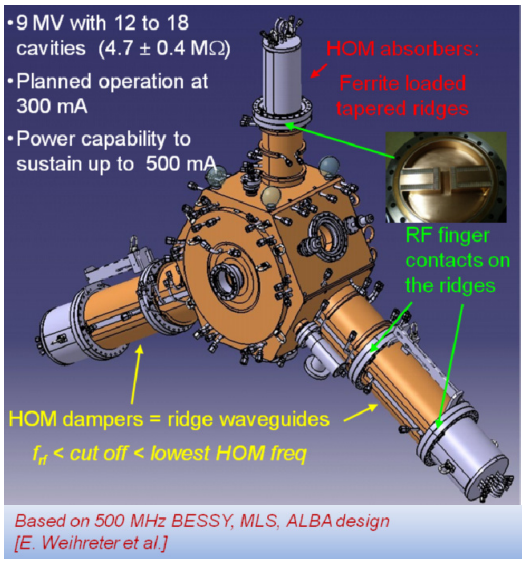


Figure 8: HOM-damped cavity design.

CONTROL SYSTEM UPGRADE

The first goal for the control system is the smooth integration of all X-ray source improvements without losing the reliability achieved over the last years.

In addition, specific long term goals have been defined for the control system.

Move to a Tango-only Control System

The ESRF accelerator complex is controlled with a mix of our legacy control system Taco, developed in the 1990's and our new Tango control system. Tango is a collaborative development which offers much more features, development and survey tools. For several years already we have been working on the replacement of Taco by Tango.

The Upgrade Programme and the long shutdown period from the end of 2011 until May 2012 offer a good opportunity to progress on the goal to replace Taco completely by Tango. The software redesign of several sub-systems is in progress to be ready for a long testing period. The exchange of the vacuum control software will be finished, by the end of 2011 and the new software for frontend and insertion device control is progressing.

A software redesign means a complete review of all features from the hardware interface to the graphical user interface.

In 2010 we still had 45% Taco devices. In 2012 we would like to have less than 25%.

Increase the Reliability

Several measures have been taken to increase the reliability of the control system.

A survey of all control computers was set up with NAGIOS [5] so that any CPU, memory or disk problem, for example, can be detected automatically.

The usage of the Tango administration system allows an immediate overview of all device servers running on control system hosts. With the new failure statistics, we can even identify infrequently occurring software crashes and schedule the necessary action.

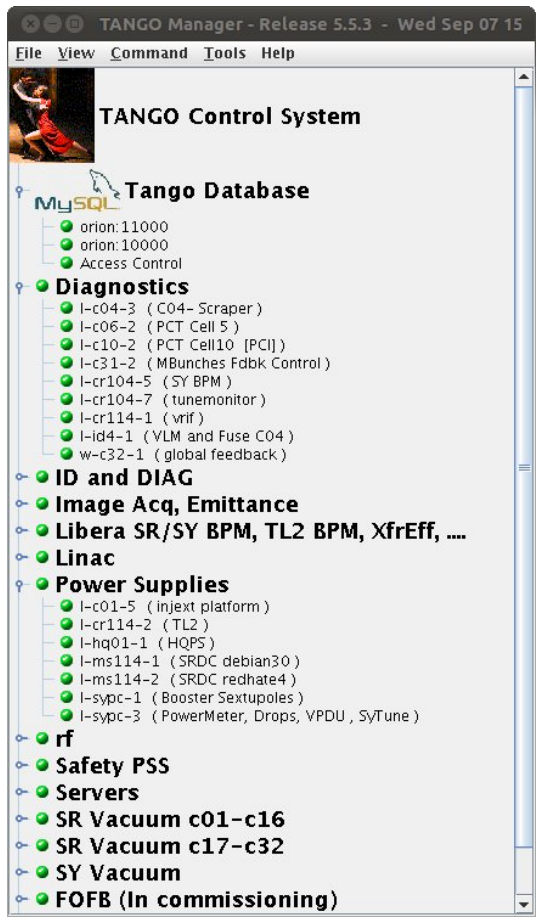


Figure 9: Tango administration GUI.

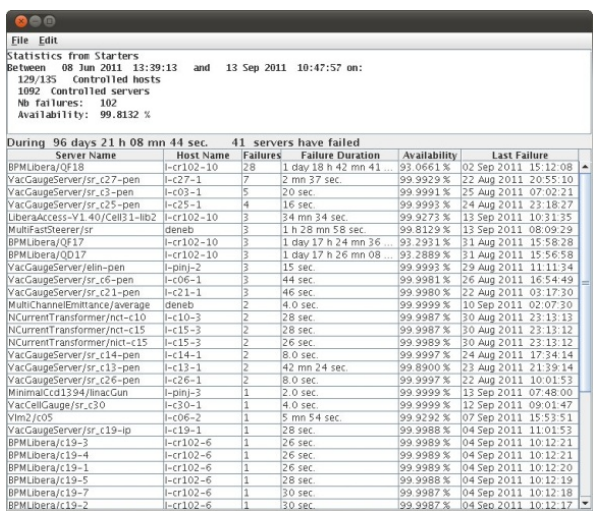


Figure 10: Failure statistics.

More High-level Analysis Tools

To allow better diagnostics or prediction of problems on the accelerator complex, a lot of effort goes into high-level data analysis tools. These analysis features can be implemented in the different device server layers, but must be regrouped into specific graphical user interfaces (GUI).

One example of such a high-level data analysis tool is the vacuum leak detection system, based on the residual gas analysers (RGA) [6], installed on the storage ring. The goal is to detect air leaks, water leaks and any abnormal out-gazing.

For example, the relative partial pressure survey is a 3-dimensional representation of selected partial pressures. Each relative pressure corresponds to a pressure change against its own reference. There is one reference pressure for each mass for every residual gas analyser and for every storage ring filling mode. This is particularly useful for a first and quick analysis of vacuum events. Figure 11 shows a typical partial pressure change signature in the case of an air leak event. Air leaks are clearly linked to strong relative changes of partial pressures corresponding mainly to masses 14, 28 and 40 (N, N2, and Ar).

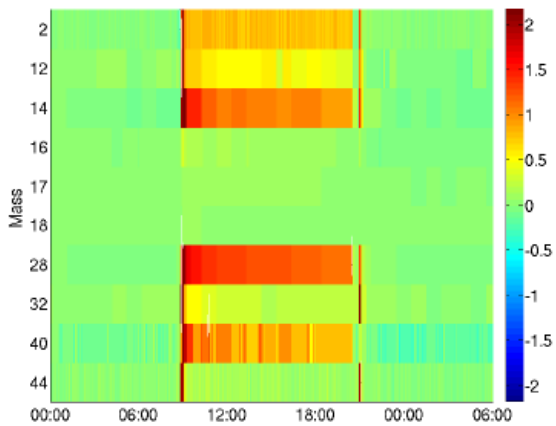


Figure 11: Real-time relative partial pressure survey.

The specific graphical user interface enables the handling of all RGAs around the storage ring and provides a complex alarm configuration as well as online and post-mortem data analysis.

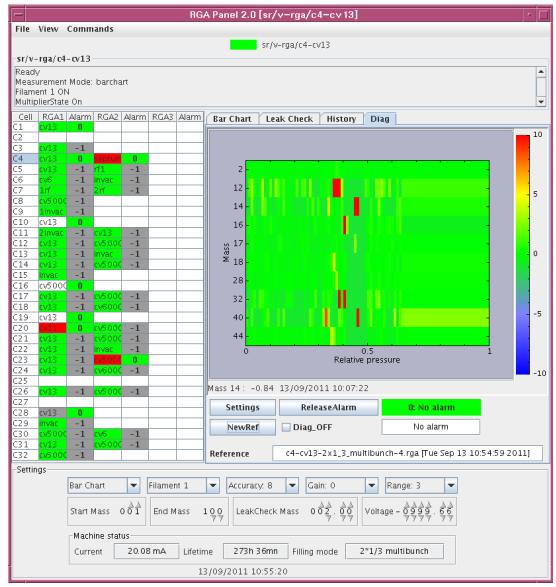


Figure 12: RGA monitoring GUI.

CONCLUSION

Conducting the upgrade in parallel to full user operation and maintaining the high stability and reliability of the X-ray source is very demanding.

As a consequence of the construction work of the experimental hall extension, a long shutdown period of 5 months is scheduled, at the beginning of 2012. During this period a lot of modifications of hardware and software will be prepared and installed. The challenge is to restart the accelerator complex with the same reliability in May 2012 to continue user operation.

We hope that all the different modifications and improvements on the X-ray source as well as on the control system will lead to a successful implementation of the new nano-focus beamlines.

REFERENCES

[1] http://www.esrf.fr/AboutUs/Upgrade
[2] http://www.tango-controls.org/
[3] http://www.i-tech.si/accelerators-instrumentation/libera-brilliance
[4] V. Serrière, J. Jacob, B.Ogier, D.Boilot, “252.2 MHZ HOM Damped Normal Conducting ESRF Cavity: Design and Fabrication”, IPAC'11, San Sebastian, Spain, 2011, http://www.JACoW.org
[5] http://www.nagios.org/
[6] A. Meunier, I. Parat, J.-L. Pons and M. Hahn, “Upgrade of the ESRF RGA System”, IPAC'11, San Sebastian, Spain, 2011, http://www.JACoW.org

CHANGING HORSES MID-STREAM: UPGRADING THE LCLS CONTROL SYSTEM DURING PRODUCTION OPERATIONS*

Sonya Hoobler, Ron Chestnut, Sergei Chevtsov, Thomas Himel, Karen Dayle Kotturi, Kristi Luchini, Jeff Olsen, Sheng Peng, Judith Rock, Robert Sass, Till Straumann, Robert Traller, Greg White, Michael Zelazny, Jingchen Zhou, SLAC National Accelerator Laboratory, Menlo Park, CA 94025, U.S.A.

Abstract

The control system for the Linac Coherent Light Source (LCLS) began as a combination of new and legacy systems. When the LCLS started operating, the bulk of the facility was newly constructed, including a new control system based on the Experimental Physics and Industrial Control System (EPICS) framework. The Linear Accelerator (Linac) portion of the LCLS was repurposed for use by the LCLS and remained controlled by a legacy system, built 25 years ago. This legacy control system is being upgraded to EPICS during LCLS production operations while maintaining the 95% uptime required by the LCLS users. The successful transition is made possible by thorough testing in sections of the Linac that were not in use by the LCLS. Additionally, a system was implemented to switch control of a Linac section between new and legacy control systems within 10 minutes. Rapid switching enables testing during maintenance periods and accelerator development days. If any problems are encountered after a section has been switched to the new control system, it can be quickly switched back. At this time, 50% of the Linac sections are operating under control of the new EPICS system. This paper describes the system upgrade with emphasis on the deployment scheme.

BACKGROUND

Hybrid Control System

The LCLS has a combination of control systems: a newly built EPICS control system and a legacy control system. The legacy system was built for the SLAC Linear Collider (SLC) 25 years ago and has been continuously improved since. The system uses distributed RMX microcomputers ("micros"), each supporting up to 15 CAMAC crates, to monitor and control accelerator components. There is a centralized host, an Alpha 6600 running the VMS operating system, which coordinates the micros, maintains a centralized database, and handles all operator interface functions [1]. Two years into LCLS user operations, this system is still used to control some functions in the Linear Accelerator (Linac) area of the LCLS. Specifically, it supports the accelerating RF system and its timing, various diagnostic signals, and the operator interface to the safety systems. The LCLS Linac is split into 11 areas, called sectors. Each sector contains a micro and four or more CAMAC crates that interface to accelerator components. Most sectors are very similar to one another and contain identical signals and devices.

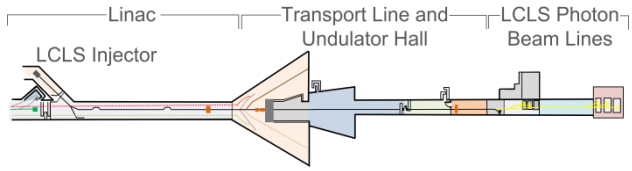


Figure 1: Overview of LCLS. [2]

Legacy System Data Access

The primary means for operators and physicists to interact with the legacy control system is a graphical user interface called the SLC Control Program (SCP). In more recent years, two additional interfaces have been developed that provide access to legacy system variables from the EPICS control system. The first is the SLC Channel Access Server (SLCCAS) [3], which provides read access to many variables over Channel Access (CA). The second is Accelerator Integrated Data Access (AIDA) [4], which provides read and write access to many variables from Matlab, Java programs, and scripts.

Upgrade Project

In order to improve LCLS uptime and reliability, dependencies on the legacy control system are being removed. In the LCLS Linac this effort has two phases. The first phase replaces the SLC micros with EPICS IOCs, fully integrating the CAMAC-controlled devices into the LCLS EPICS control system. The second phase will upgrade the CAMAC to new hardware. This paper discusses the first phase.

Challenges

The legacy control system has benefited from decades of refinement and is feature-rich. Additionally, thousands of signals in the LCLS Linac must be ported to EPICS. Reproducing features and accurately copying all signals are two main challenges of this project. Another challenge is to perform the upgrade during a production LCLS run, while meeting strict uptime requirements for LCLS users.

*Work supported by U. S. DOE Contract DE-AC02-76SF00515

DESIGN

System Architecture

Each Linac micro is being replaced by a standard LCLS EPICS IOC, which consists of an MVME6100 processor running the RTEMS operating system installed in a VME crate. A serial CAMAC driver board was developed to enable an IOC to communicate with CAMAC hardware. EPICS CAMAC drivers, EPICS databases, and higher-level IOC software were also created to replace the legacy software.

A switching interface chassis was built for each sector. Each chassis determines whether that sector is controlled by a legacy micro or a new IOC. Control can be switched locally or remotely. Thus some sectors can be controlled by the legacy system while others are controlled by EPICS, and sectors can be tested or upgraded independently. The switching chassis also greatly aided testing, allowing developers to compare live data between the IOC and micro by switching back and forth.

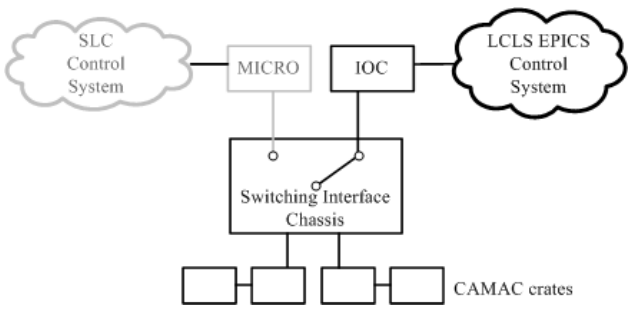


Figure 2: Hardware to enable switch between micro and IOC control.

Automated Data Transfer

Thousands of signals were ported to EPICS. To reduce errors, scripts were written to transfer and update data. For example, scripts were used to:

- Generate EPICS databases from the SLC database.
- Copy variable values to the new system when switching a sector to EPICS control. Variable values were not automatically copied from EPICS to the legacy system when switching back. This provided a known good configuration to return to in case of problems, with the disadvantage that when switching back to the legacy system, some variable values had to be copied manually.
- Modify existing user displays, replacing old PV names with new names.

Application Programming Interface

Before the upgrade, many high-level software applications used AIDA to read and control RF system variables. These applications did not need to be modified. This is because AIDA was updated to read and control these variables through EPICS as well. Additionally, the AIDA directory service is updated when a sector is switched over and thus AIDA accesses the data from the appropriate control system. Applications that used Channel Access to access SLCCAS data had to be updated with the new PV names.

When converting SLC database variable names to EPICS Process Variables (PVs), some names were changed to comply with the LCLS naming conventions.

UPGRADE TO EPICS

Development Testing

Initial testing was performed in a lab test stand with two CAMAC crates. Fortunately, two unused Linac sectors were available for several months of the initial testing, on which much of the software could be tested. These sectors' RF systems were not all available, however, so not all software could be thoroughly tested there.

Transition Infrastructure

In parallel with the development testing, some infrastructure was released to production to support and ease the transition, and to facilitate the switching of individual sectors. These were done months before the upgrade:

- PV gateway: A PV gateway was installed to translate the old SLCCAS PV names to the future EPICS PV names [5]. Scripts were written to reconfigure the gateway, based on which system was controlling each sector. This gateway allowed clients to begin using the new names early for read-only purposes.
- Data archiving: PVs were added to the Channel Archiver early in the project so that control system users would have access to months of data history when sectors were switched over. The users agreed that this provided sufficient archived history in the new system and that they would use the old system if they needed older data.
- User interfaces: New and existing EDM screens, using the new PV names, were released to production. The original displays remained available for comparison or as a backup—in case users suspected problems with the new displays.
- Alarms: Signals were added to the alarm handler. On a per-signal basis, these alarms were automatically disabled if the associated sector was not under EPICS control.
- Data sources: Scripts were written to update the AIDA directory service when a sector was switched.

Switchover Tool

Scripts were created to perform most of the tasks associated with switching a sector. Using these scripts, switching a sector took less than 10 minutes. These scripts were made available on operator displays, so operators could switch a sector back to the legacy system if needed. This happened on several occasions, when operators needed to use a feature that had not yet been implemented in the new system or to compare data in the new system with the familiar legacy system. This made the control system users more comfortable with leaving a sector under EPICS control, knowing they could easily back out if needed. This also made it convenient to test for a relatively short period. These scripts did the following:

- Reconfigure and restart the PV gateway
- Update AIDA directory service
- Boot IOCs
- Copy settings over (one way)
- Restart some Channel Access clients

Production Testing

In production, testing was done on the LCLS Linac during scheduled maintenance periods and accelerator development days. As individual sectors could be switched, it was sometimes possible to test a sector that was not needed for that day's accelerator program. Most of the tests lasted a few hours. The sector was switched to EPICS control before testing and back to the legacy system afterward. These short tests provided opportunities to find problems and then fix them without using valuable beam time. Adding all the time up, testing required about 25 hours during maintenance periods and for about 60 hours during physics studies. Further testing took place during a 2-month scheduled downtime when all sectors were switched to EPICS control. This allowed maintenance groups to become familiar with the new system, although it did not allow extensive testing as most systems were off.

PRESENT STATUS

At the present time, about half of the sectors are under EPICS control. Small bugs are being fixed and remaining features are being completed. The remaining sectors will be switched to EPICS in the next month or two. This design is expected to be used when upgrading other (non-LCLS) areas of the SLAC Linac.

CONCLUSIONS

Extensive testing in a lab and in unused Linac sectors greatly aided development. The ability to switch between control systems and compare data was found to be invaluable during testing.

Great effort was taken to create an implementation scheme with minimal disruption to the user program. This was found to be worthwhile. To date, interruption to the user program has been less than 30 minutes.

REFERENCES

[1] J. Bogart et al., “SLC Control System Basic Users Guide,” April 1995; http://www.slac.stanford.edu/grp/cd/soft/wwwman/bug.www/

[2] L. Christensen, “LCLSI Beamline Overview”, SLAC National Accelerator Laboratory, October 2011

[3] S. Allison, “SLC Channel Access Server Software Design Specification”, March 1998; http://mccdev.slac.stanford.edu/doc$spec/SLC_CA_SERVER.PDF

[4] G. White, “AIDA Acceleratator Integrated Data Access”; http://accelconf.web.cern.ch/accelconf/icalepcs2009/papers/thp020.pdf

[5] D. Kotturi, “Quick Changeover Tool Software Design Specification”, November 2009

NEW DEVELOPMENT OF EPICS-BASED DATA ACQUISITION SYSTEM FOR MILLIMETER-WAVE INTERFEROMETER IN KSTAR TOKAMAK

Taegu Lee, Mikyung Park, and Y. U. Nam
National Fusion Research Institute (NFRI), Daejeon, Korea

Abstract

After successful achievement of the first plasma in 2008, Korea Superconducting Tokamak Advanced Research (KSTAR) has performed the 4nd campaign in 2011. During the campaigns, many diagnostic devices have been installed for measuring the various plasma properties in the KSTAR tokamak. From the first campaign, a data acquisition (DAQ) system of Millimeter-wave interferometer (MMWI) has been operated to measure the plasma electron density. The DAQ system at the beginning was developed for three different diagnostics having similar channel characteristics with a VME-form factor housing three digitizers in Linux OS platform; millimeter-wave interferometer, H-alpha monitor and ECE radiometer. However, this configuration made some limitations in operation although it had an advantage in hardware utilization. It caused unnecessarily increasing data acquired from the diagnostics when one of them operated at higher frequency. Moreover, faults in a digitizer led to failure in data acquisition of the other diagnostics. In order to overcome these weak points, a new MMWI DAQ system has been developed with a PXI-form factor in Linux OS platform and a control application has been developed on EPICS framework like other control systems installed in KSTAR. It also includes MDSplus interface for the pulse-based archiving of experimental data. Main advantages of the new MMWI DAQ system besides solving the described problems are capabilities of calculating plasma electron density during plasma shot and displaying it in run-time. To this end, the data can be provided to users immediately after archiving in MDSplus DB.

INTRODUCTION

In the first campaign of KSTAR, only a few diagnostics including the millimeter-wave interferometer were involved in experiments, but the number of diagnostics continuously increased according to the diagnostics upgrade plan and finally almost 30 different diagnostics were operated in the 4th campaign. Among many plasma physics parameters, electron density is a primary parameter and an interferometry is a widely used diagnostic tool for a measurement of the electron density [1][2]. The first DAQ system for the interferometer was developed with a VME-form factor housing three digitizers in Linux OS platform which was shared by three different diagnostics having similar channel characteristics; millimeter-wave interferometer, H-alpha monitor and ECE radiometer. Also, the DAQ system was developed on EPICS [3] middleware like other plant control systems and diagnostic DAQ systems installed in KSTAR, and performed as a standard input output controller (IOC). However, it turned out in the operation of last campaigns that the interferometer DAQ system made some limitations although it had an advantage in hardware utilization and in terms of maintenance. It caused unnecessary increase in the amount of acquired data when one of three diagnostics operated at higher frequency and resulted in a longer waiting time to read data stored in MDSplus [4] DB [5] Moreover, faults in a digitizer led to failure in data acquisition of the other diagnostics. In order to overcome these weak points and to increase the number of diagnostic channels, a dedicated DAQ system for the ECE radiometer was first developed before the 3rd campaign in 2010. Next, a Millimeter-wave interferometer (MMWI) DAQ system has been newly developed to measure electron density for the experiment of the 4th campaign in 2011 (see Table 1).

The MMWI DAQ system has been developed with a different H/W form-factor, PXI, and achieved more accurate timing and operation availability by adopting a new timing board and a standard software framework. Particularly, it can display raw data and calculated electron density data at once during plasma shot in real-time, and this feature is useful for longer pulse plasma operation. Same as the previous system, all the interferometer data are archived using MDSplus, while some of data is shared via EPICS Channel Access (CA) and displayed by using EPICS strip. The OPerator Interface (OPI) for the MMWI was developed using KWT, which stands for KSTAR Widget Toolkit and is a set of libraries developed in house using Qt [6].

Table 1: Channel Configuration in DDS#2 [5] (Old DAQ System)

Diagnostics	2008	2009	2010	2011
MMWI	4 ch	4 ch	4 ch	New DAQ (2 ch)
H-Alpha	30 ch	30 ch	30 ch	30 ch
ECE	8 ch	40 ch	New DAQ (48 ch)	76 ch

UPGRADE OF MMWI DATA ACQUISITION SYSTEM

The new MMWI DAQ system has been developed with PXI form-factor digitizers in 32-bit Linux OS platform to solve the described problems and improve performance.

The PXI system should use the NI-DAQmx library to interface EPICS device/driver in Linux OS. NI-DAQmx library supports only x86 32-bit architecture for Linux and support only some of NI devices.

H/W Upgrade

The MMWI data acquisition system consists of a 2.53Ghz dual-core PXIe embedded controller running on Linux,a home-made Local Timing Unit (LTU) and a Analog-to-Digital Convertor module[7]. The NI (National Instrument) PXI-6123 ADC module has 8 analog input channels, 16bit resolution, 4 configurable input ranges from ±1.25 to ±10V, and 64MB onboard memory. Although the input signal from a phase comparator to the ADC module varies from 0 to 5V, the operating input range is ±10V in order to avoid input saturation. The sampling rate in normal operation is 100kS/s per-channel. The bit-resolution of the new MMWI DAQ system was downgraded from 24bit to 16bit but the sampling rate was upgraded from 216kS/s to 500kS/s.

The MMWI data acquisition sequence is synchronized with KSTAR experimental cycle using the LTU and a dedicated timing network like other installed diagnostic DAQ systems. The LTU is configured with a pre-programmed scenario before plasma shot, and generates triggers and clock signals according to the scenario as soon as it receives a shot start signal from a Central Timing Unit (CTU) thru the optical timing network [8].

Specifications of the new MMWI DAQ system are as followings:

- NI PXIe-8108: 2.53Ghz dual-Core PXI Express, 4GB DDR2 RAM, up to 1GB/s system bandwidth and 250 MB/s slot bandwidth.
- NI PXI-6123: 8 simultaneously sampled analog inputs, 16-bit resolution, max 500 kS/s per channel, input ranges from ±1.25 to ±10 V.
- NI BNC-2011: BNC connectors for analog I/O, Shielded enclosure.
- LTU: accuracy < 5ns, resolution 5ns, output clock (1Hz ~ 200MHz), max 8 configurable multi triggering sections 2Gbps Optical communication speed, max 8 Trigger/Clock outputs

Figure 1 shows the configuration of the new MMWI DAQ System and network connections to the infrastructure of the KSTAR control system.

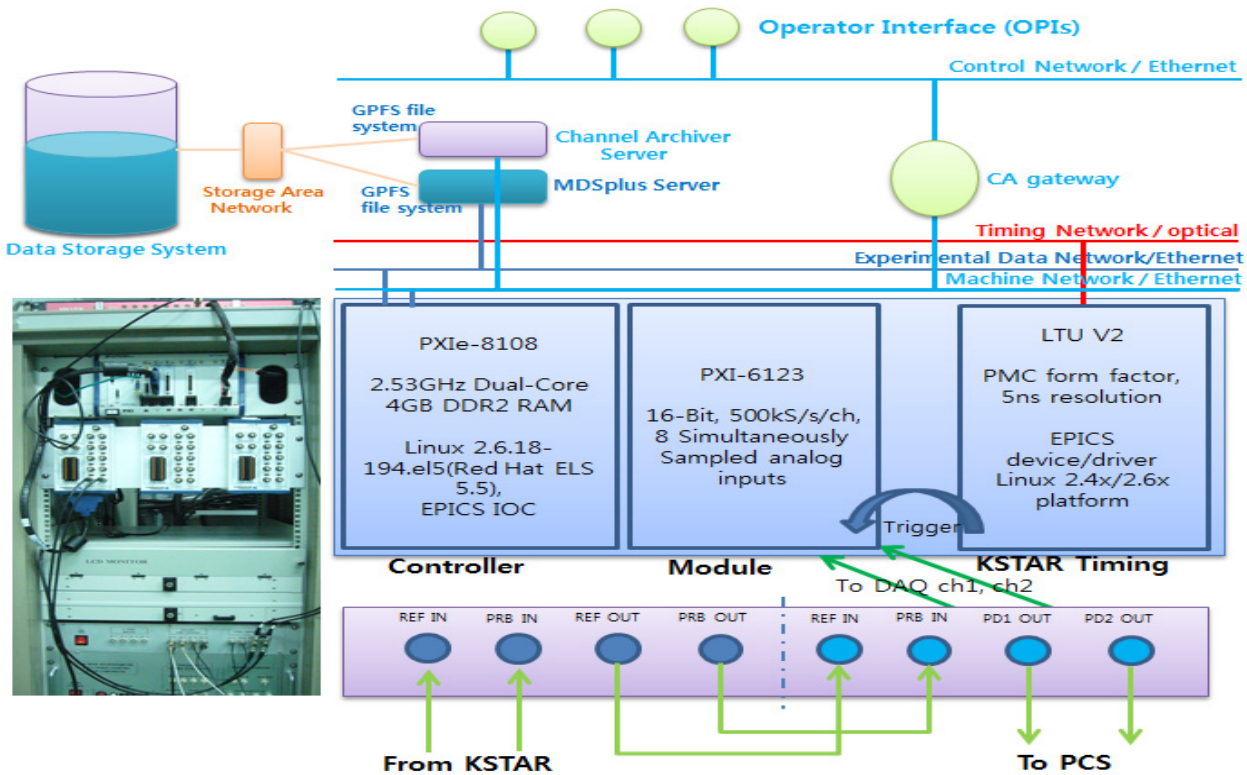


Figure 1: Configuration of MMWI DAQ System and network connections to the infrastructure of the KSTAR control system.

S/W Development

As mentioned above, the MMWI DAQ system is running in Linux OS platform, RedHat ES 5.5 (kernel – 2.6.18-194.e15). In general, all the data acquisition systems in KSTAR adopt a standard Software Frame-Work (SFW) that is developed and continuously improved in-house to reduce development time and improve system reliability. The SFW utilizes the IOC core software provided by EPICS and includes driver and device support software in common used with many other data acquisition systems in KSTAR [9]. The EPICS device/driver of the MMWI DAQ system is based on NI-DAQmx library to control National Instruments devices in Linux OS.

The SFW provides essential functions to support common records (shot number, blip time, seven states of system condition, etc), provide commonly used MDSplus interface to save acquisition data sampled at a high rate, and synchronize with a KSTAR plasma operation mechanism [9]. The OPI panels are developed by a particular set of Qt libraries named as KSTAR Widget Toolkit (KWT). They comprise of several sub-panels to configure and operate data acquisition, and to display the two types of density data in run-time; density data at 1 sec rate in AI record and full density data in a buffer in Waveform record. Additionally, the MMWI DAQ system uses common libraries of the KSTAR control system; the timestampLib to provide system time information and the sysMonLib to provide system health status of the system. Figure 2 shows the function blocks of MMWI DAQ IOC.

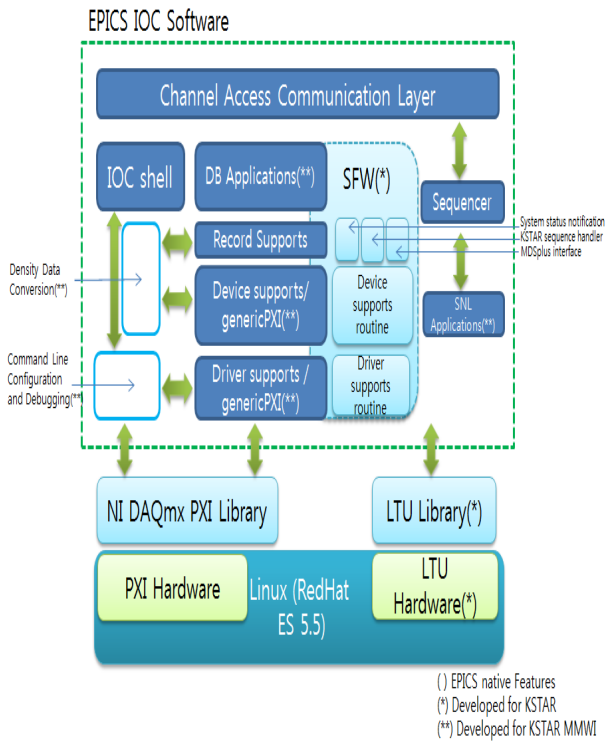


Figure 2: Function blocks for MMWI IOC.

All the programs to operate the MMWI DAQ system and archive diagnostic data are summarized as hardware driver for PXI devices (NI-DAQmx8.0.2), EPICS middleware interface program, MDSplus interface program, and Graphical User Interface (GUI) program (see Table 2).

Table 2: Environment for Software Development

Program	Version	Site
EPICS	3.14.12	http://epics.aps.anl.gov/epics
MDSplus	2.3-0	http://www.mdsplus.org
Qt	4.3.2	http://www.trolltech.com
DAQmx	8.0.2	http://www.ni.com

Calculation of Plasma Density for Real-Time Service

The millimeter-wave interferometer is a diagnostic tool to measure plasma density by using the phase changes according to density in the propagation through plasma. When millimeter-wave travels through plasma, its phase varies depending on plasma density, and the phase difference from phase through vacuum is detected and converted to voltage signal to be proportional to the measured phase by a phase comparator. If the measured phase exceeds 2π, the fringe jump phenomenon occurs and the output voltage goes back to zero. In order to provide the density value to users in real-time, the MMWI DAQ system writes the density data in AI record and Waveform record after correcting fringe jump with two raw data having the phase difference of π.

When "DMA done interrupt" from the NI device is received at every one second, the IOC first performs a fitting algorithm to compensate for a slight curvature of the phase comparator circuit. Next, the IOC averages 1000 raw data from each of two input channels and then judges which signal is closer to the center of maximum detected voltage to select better channel for calculation of density. The averaging is executed once at the start of a DMA done event.

Even though the input channel is selected, it is changed to the other to avoid the fringe jump if it's voltage gap from the center voltage is larger than the other channel. When DMA data processing is complete, the IOC saves one calculated density data and two raw data from two input channels (without fitting) to files in the local HDD. It updates the measured density value in the waveform record and the final data in the AI record. This sequence is repeated at every DMA interrupt. After the completion of plasma shot sequence, the IOC moves two raw data and density data saved in the local HDD to MDSplus DB in the KSTAR central storage. Figure 3 shows the entire sequence of density calculation with the fringe-jump correction algorithm in IOC.

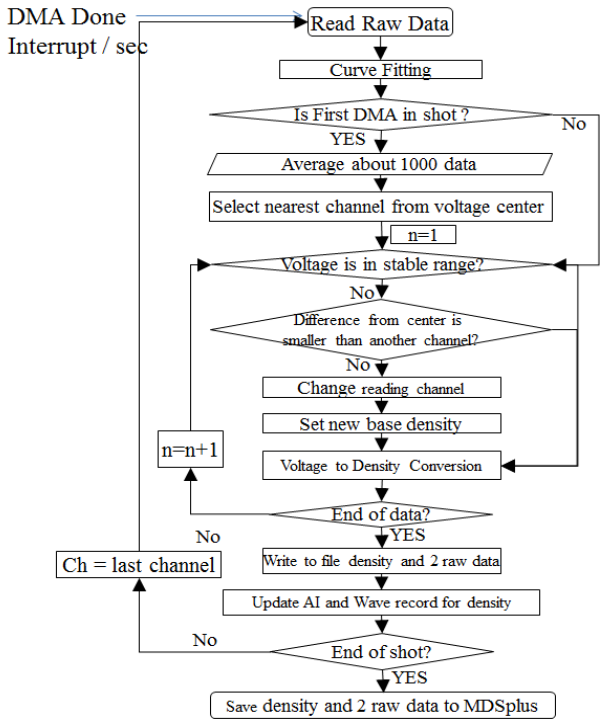


Figure 3: Sequence of density calculation with fringe jump correction algorithm.

OPERATION RESULT IN THE 4TH CAMPAIGN

After upgrading the MMWI DAQ system, both the stability and the usability of the system has increased considerably, and the amount of unnecessary data has also reduced. As shown in Table 3, just one fault occurred during the 4th campaign and even it was cause by the problem of NI-DAQmx on Linux in the case of long time operation. This problem is a big obstacle for a long time operation without system reset. Moreover, the same problem has occurred in all NI DAQ systems in KSTAR since 2008, and become the most significant fault source in the operation of data acquisition systems. We have now focused on solving it in cooperation with the manufacturer.

Table 3: Fault Counts in the Operation of the MMWI DAQ System

Campaign	DAQ fault counts	Lost shot	Total shot
1st 2008	23	23	1283
2nd 2009	4	2	1059
3rd 2010	14	17	2126
4th 2011	1	2	2002

Another deficiency is a small number of data points displayed in run-time during a shot. It displayed only 10 points data during a plasma pulse which was max 10sec in length in 2011. In the plasma experiments period, the new DAQ system operated successfully in data acquisition and density calculation only except for the case that the density changed momentarily and rapidly at the high density plasma. In addition, the heavy noise induced in the DAQ system has not been settled down.

SUMMARY AND FUTURE PLAN

In the fourth operation of KSTAR in 2011, the newly developed MMWI DAQ system was built and operated as an independent system. Add to the features of the previous system, the new DAQ system has many advantages in the views of hardware and software; the improved performance in data acquisition by adopting the standardization, more accurate synchronized operation with a new timing board, the run-time calculation and displaying density data. Also, there was a progress in the efficient data management. In the next campaign, the MMWI DAQ system will be modified to meet requirements arising in operation such as increasing the sampling counts to 10 for the effective run-time displaying and adding the function to archive the calculated data into MDSplus in real time to reduce the waiting time. Also, we will solve the problem of NI-DAQmx in the cooperation with the manufacturer.

REFERENCES

[1] Y.U.NAM, and K.D.LEE, Rev. Sci. Instrum. 79, 10E705 (2008).
[2] Y.U.Nam, Rev. Fusion science and technology. ISSN 1536-1055 (2009).
[3] http://www.aps.anl.gov/epics.
[4] http://www.MDSplus.org.
[5] Mikyung Park, et.al.., "Data acquisition system implemented with various platform for KSAR diagnostic systems", Fusion Eng.Des. 85 (2010) 350-355.
[6] S.Baek, et.al.., "KSTAR widget toolkit using Qt library for the EPICS-based control system" Proceedings of ICALEPCS (2009).
[7] Mikyung Park, et al.., "Development of a Prototype Time Synchronization System for KSTAR with a VME-bus System", Fusiion Eng. Des. 81 (2006) 1817-1821
[8] Mikyung Park. Other, "The Upgrade of KSTAR Timing System to Support Long Pulse Operation and High speed Data Acquisition", Proceedings of IAEA TM (2011).
[9] Woongryol Lee, Other, "Development of the EPICS-based data acquisition system for Thomson scattering diagnostic", Proceedings of Fusion Eng.Des (2011).

EVENT-SYNCHRONIZED DATA ACQUISITION SYSTEM OF 5 GIGA-BPS DATA RATE FOR USER EXPERIMENT AT THE XFEL FACILITY, SACLA

Mitsuhiro Yamaga*, Arnaud Amselem, Toko Hirono, Yasumasa Joti, Akio Kiyomichi,
Toru Ohata, Takashi Sugimoto, Ryotaro Tanaka, JASRI/SPring-8, Hyogo, Japan
Takaki Hatsui†, RIKEN/SPring-8, Hyogo, Japan

Abstract

We have developed a data acquisition, control, and storage system for user experiments at the X-ray Free Electron Laser facility, SACLA, at the SPring-8 site. The system is designed to handle up to 5.8 Gbps data rate of shot-by-shot data in synchronization with the 60-Hz of beam operation cycle. A partial system for the light source commissioning was released in March 2011. The full system will be released to public users in March 2012.

INTRODUCTION

SPring-8 Angstrom Compact Free Electron Laser (SACLA) has been constructed at the SPring-8 site. SACLA is designed to generate X-ray laser with a wavelength as short as 0.062 nm by using a thermionic electron gun, an 8-GeV linear accelerator, and in-vacuum undulators [1]. The accelerator commissioning began in February 2011. The first lasing at a wavelength of 0.08 nm was achieved in June 2011.

The characteristics of XFEL such as the high-peak brilliance, perfect spacial coherence, and the ultra-fast X-ray pulse shorter than 30 fs are believed to open new opportunity in a variety of scientific fields. SACLA will deliver beam time to public users from March 2012.

DATA ACQUISITION SYSTEM FOR USER EXPERIMENT

Requirements and System Overview

Because the SACLA accelerator is driven with the pulsed operation, the X-ray laser is generated accordingly as the pulsed beam. The X-ray laser is generated by Self-Amplification of Spontaneous Emission (SASE) resulting in the fluctuation of the X-ray laser characteristics such as in the pulse intensity, spectrum, etc. In order to analyze the experimental results taking these fluctuation into account, the data acquisition (DAQ) in the experiment at XFEL must be done shot-by-shot in synchronization with the beam operation cycle. Furthermore, in many of the anticipated experiments, a single shot of the X-ray pulse will damage the sample specimen, the synchronized DAQ is indispensable to correlate the recorded data with the specimen characteristics such as orientation, size etc.

* email: myamaga@spring8.or.jp
† email: hatsui@spring8.or.jp

The most experiments will produce data with one-dimensional (1D) data such as waveform from digitizers, two-dimensional (2D) data from optical camera or multi-port charge-coupled device (MPCCD) for X-ray imaging. The devices installed at SACLA produce the data with a size ranging from 0.5 to 10 MB for each X-ray pulse. With 12 MPCCD sensors operating simultaneously at the 60-Hz operation cycle, which is the maximum number of sensors for the first user-experiment runs, data rate reaches 5.8 Gbps. The beam lime optical components consist of mirrors, a monochromator, attenuators, and beam monitors for intensity, position and arrival time. The control system should be designed so that users can correlate component status as well as monitor data with other detector data.

In order to meet these requirements, we have developed a data acquisition (DAQ) system for SACLA. The DAQ system has dedicated networks to transmit the data from all the instruments, monitors, and detectors. The system is also indirectly connected to the user network via a firewall system so that user can attach their own instruments to operate in a coordinated manner. This will lower the cost in building new experimental setup, and add flexibility to the DAQ system.

The DAQ system has on-line monitoring capability for the prompt examination of the experimental conditions. Raw and preprocessed data can be visualized by LiveView service operating in data-handling servers. In the case of the experiments that demand data preprocessing or data mining by high-performance computer for on-line data evaluation, the DAQ system can transfer the data to a PC cluster with 10 Tflops with high I/O performance (8 GB/s), or to the off-site 10 Pflops supercomputer "K computer".

The on-the-fly data compression relaxes the requirements of the bandwidth of data transfer as well as the final storage size to accumulate the experimental data. Embedded data compression for 2D data is available at the front-end Camera Link boards.

Control Framework

For control of the facility equipments, we have selected the MADOCA software framework [2], which has been developed and operated at the SPring-8 accelerator complex. The network-distributed computers are controlled via an RPC call by message-passing system. The system consists of the trigger distributors/counters, data-filling computers, parallel-writing high-speed data storage, and relational databases. Figure 1 shows the schematic of the data-

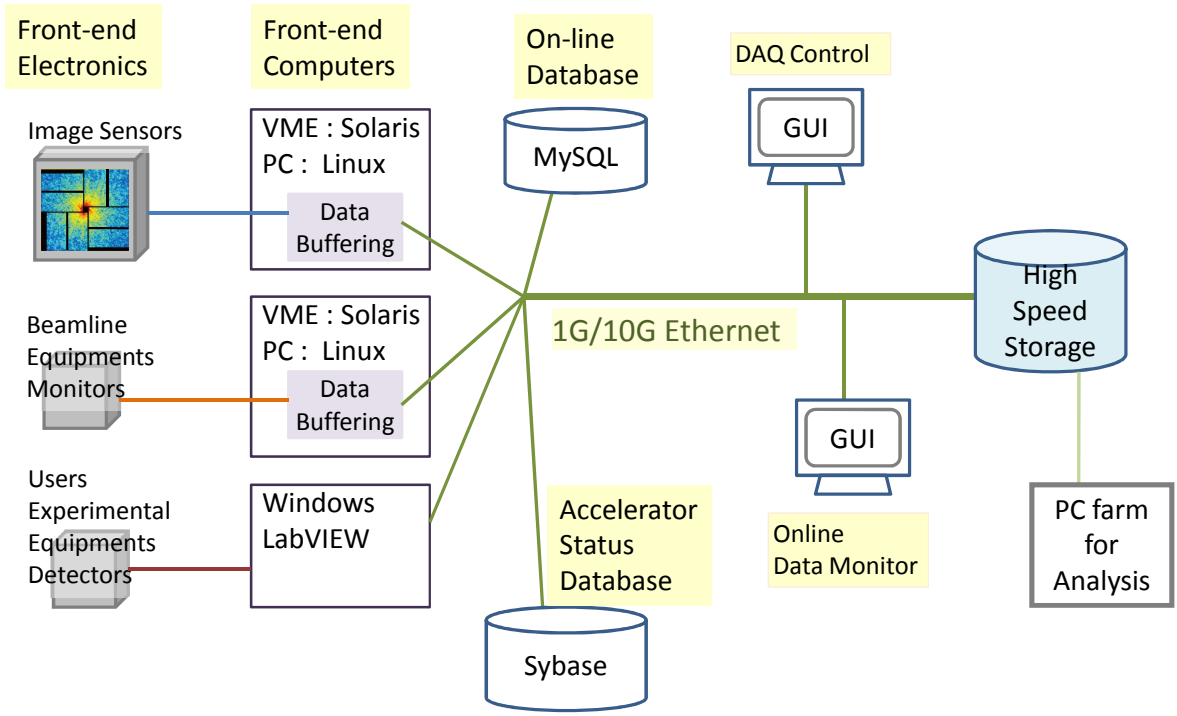


Figure 1: Schematic of the data-acquisition system for SACLA user experiment.

acquisition system for SACLA user experiment.

The original MADOCA has a limitation in the data transfer size. In order to enable a large data transfer for SACLA experiments, a client-server system with TCP socket is implemented to the framework. The large data set is transferred directly via TCP socket system, while the client and server process are controlled by the MADOCA messages. We have demonstrated that 80 % bandwidth of 1 Gbps Ethernet is available for the peer-to-peer data transfer with this system within the modified MADOCA framework.

We introduced the Qt software as the graphical user interface (GUI) for the control tools and data monitoring tools of the facility equipment. The Qt designer software allow us the prompt prototyping of the design of the GUI. The C++ program code of GUI is convenient to call the C API function of MADOCA directly from the GUI itself. So far more than 10 GUIs have been created for our DAQ.

Connectivity to the other software development environment became possible by using the TCP socket data transfer function added in this work. To date, these new functionalities yielded in successful operation of the real-time detector-performance-analysis software implemented on commercial software Igor Pro on Windows platform[3], and of an apparatus to measure the focused X-ray beam running on LabVIEW software on Windows platform.

Front-End System

The most critical part of our DAQ is to collect data from large 1D and 2D data without data loss in synchronization with 60-Hz beam operation cycle.

In the first experimental runs starting from March 2012, we plan simultaneous use of maximum two arrayed MPCCD detectors comprising totally 12 sensors. Each sensor has 0.5 M pixels in size with 16-bit quantized depth. A total throughput of 5.8 Gbps will be required for this sensor. We have chosen Camera Link, which has a transfer rate of more than 2 Gbps, for the interface of the image sensor. The required throughput is achieved by a configuration where each sensor has one Camera Link base-configuration interface. We use Camera Link connections independently for 12 sensors for the convenience of the later data handling. By utilizing an industrial-standard Camera Link interface, quick launch of the data acquisition system was achieved by simply applying a commercial image grabber board on the PC Linux system.

We have also developed a custom Camera Link interface board as a VME module as an upgrade of image acquisition. The module converts the Camera Link protocol to the Ethernet protocol so that the data are directly sent via the network from the front-end VME module to the back-end computers. An on-the-fly data-compression algorithm is implemented in the FPGA on the board so that the data size could be reduced at a very early stage of data acquisition. In the SACLA experiments, decompression speed on CPUs is one of the important performance of compression algorithm. We have developed and implemented a modified Range-Coder compression algorithm on FPGAs, which has superior performance in decompression speed than other standard lossless compression algorithm such as

JPEG 2000. The data size is reduced down to 40-50 % of its original size.

The waveform of the detector, such as in the time-of-flight experiment, is collected by a digitizer at the sampling rate of 4-GHz. The data size of waveform is to be up to 1 MB per X-ray pulse per detector channel before pedestal subtraction, and would be up to 100 kB after pedestal subtraction. A total throughput of up to 1 Gbps will be required. The PCI-express extension board on the PC Linux system is adopted for the digitizer to handle this data rate.

The intensity and the transverse location of the X-ray pulse are measured by monitors equipped with PIN photodiodes. The voltage waveform outputs from preamplifiers are digitized and fitted to Slater function on VME digitizers (MVD-ADC03) to deduce the charge detected by PIN photodiodes. A Solaris 10 operating system runs on the Sanritz SVA041 VME CPU board to control the ADC and other modules in the same VME chassis.

The profile of X-ray beam is observed by the YAG-Ce scintillator screen with a 0.3M pixels CCD camera system. The Camera Link interface board identical to MPCCD sensors is applied for the image acquisition of this CCD camera.

Timing System and Tag Number

The timing signal of the master-trigger system to drive the SACLA accelerator equipment at 60-Hz cycle is also delivered to the DAQ system for user experiment so that the detector could be operated exactly in synchronization with the timing of the X-ray pulse.

In the distributed DAQ system, shot-by-shot data is recorded independently at each front-end computer per X-ray pulse. Each datum must be uniquely identified so that one can collect all the data recorded in the same X-ray pulse to reconstruct the event. We assign a tag number, which is actually a sequential number of the master-trigger signal, to every datum to identify which one is related to which X-ray pulse. Because our master-trigger system does not distribute the encoded trigger number itself, the tag number is counted at each front-end computer node independently by the trigger counter module. All counters are set to the identical number at lease once at a certain point of beam operation to assign the identical tag number for the same X-ray pulse at every node.

Database

Most of the detectors for the facility equipment, such as the beam intensity monitors and beam position monitors, are always in operation when the DAQ system is ready. These data are taken shot-by-shot and are written to the fast relational database MySQL, with a tag number and the timestamp record. These data are shown as the present status of the XFEL beam on the monitoring system at the experimental station. The user can ascertain whether the beam condition satisfies the experimental requirement by this monitoring system. Users can acquire the data such as the beam condition from the database by specifying the tag number or the timestamp. For example, the correlation between the experimental data and the beam condition could be easily analyzed by fetching the data from this database. Access to the database by user programs is also possible via an interface server.

Network and Pipeline Scheme

The most important issue in the network for our DAQ is to guarantee the data transfer without loss. Therefore, in our system, all the data are transmitted via TCP/IP protocol over the standard 1- and 10- Gbps Ethernets to achieve the reliable data transfer by the features of TCP, e.g. ordered data transfer, retransmission of lost packets and error detection, with a reasonable cost.

Because the Ethernet is not capable of the critical real-time operations, we implemented a pipeline FIFO buffering system in the software framework of client-server system of TCP socket connection so that the data would not be lost at the application level even when data transfer is delayed. The FIFO buffer is implemented as a ring-buffer with shared-memory on a Solaris and Linux operating system. Owing to recent progress of information technology, a large memory size of over a gigabyte is available on the computer. We could make the pipeline depth large enough to hold more than 10 seconds of data, which would be sufficient even when the large latency of TCP/IP protocol occurs.

Storage

The 1D and 2D data from the front-ends are finally accumulated to the storage system. It must be able to handle up to 5.8 Gbps of throughput. The total size of data set will reach over 50 TB per day at this maximum rate. For ease of use, a single file system is preferred even for such a large storage.

We have chosen the StorNext single file system on the Data Direct Network (DDN) S2A9900 Storage system to satisfy these requirements. A total of 180 TB of storage is provided by the SAN cluster of the RAID system at over 1 GB/s of total throughput. Five Linux PCs are prepared as the parallel-writing file servers in this SAN system. By distributing the data from the front-end computers into these five file servers in parallel, we have demonstrated that the image and waveform data of up to 5.8 Gbps can be recorded to storage without any data loss.

PRESENT STATUS

Beam tuning of XFEL began in March 2011. During tuning, our DAQ system is intensively used to provide beam intensity data and beam profile data at the beam line components. Stable operation of our DAQ system contributed significantly to efficient and prompt beam tuning.

Several preliminary experiments are planned by internal users of our facility until March 2012. The 8-plates

MPCCD sensor will be used in these experiments. The beam repetition rate until March 2012 will be 10 Hz at maximum as a preliminary operation of SACLA. We will check the performance of the each component of our DAQ system during these preliminary experiments.

FUTURE PROSPECTS

We anticipate that a larger size image sensor will be required for the future experiments using the XFEL. In order to achieve smaller pixel size with higher peak signal and dynamic range, we are developing Multi-Via (MVIA) sensor by using SOI sensor technology [4] as the next-generation X-ray imaging sensor for SACLA. The detector size will be 1.9 M pixels per sensor with 16-bit quantized depth. By assembling 40 MVIA sensors, the total data rate will be 13 times larger than that for the first experimental runs.

An increase of the beam repetition rate from 60 to 120 Hz at SACLA is expected to be rather promising in near future with the present technique. At 120-Hz beam repetition rate, the 40 MVIA sensors will produce 146 Gbps data rate, which is 26 times higher than the present system. To meet these requirements, upgrades of the data transfer networks, front-ends, parallel implementation of the storage system are now under investigation.

SUMMARY

A data acquisition, control, and storage system has been developed for user experiments at the XFEL facility, SACLA, in the SPring-8 site. The shot-by-shot data acquisition in synchronization with the beam operation cycle of up to 60-Hz has been performed in the beam tuning of XFEL laser. The data transfer scheme with the ordinary 1- and 10- Gbps Ethernets has been proved to be reliable enough by introducing the pipeline buffering in the client-server system of TCP socket connection. We have demonstrated that image and waveform data of up to 5.8 Gbps could be recorded in parallel-writing high-speed data storage without any data loss.

REFERENCES

[1] T. Shintake et al., X-ray FEL project at SPring-8 Japan, Proceedings of 8th International Conference of Synchrotron Radiation Instrument (SRI2003), 227 (2004).

[2] R. Tanaka et. al., The first operation of control system at the SPring-8 storage ring, Proceedings of ICALEPCS97, Beijing, China, 1 (1997).

[3] Igor Pro, WaveMetrics, http://www.wavemetrics.com/.

[4] SOIPIX collaboration, Developments of SOI monolithic pixel detectors, Nuclear Instruments and Methods in Physics Research Section A, 623(1) 186 (2010) .

INAUGURATION OF THE XFEL FACILITY, SACLA, IN SPRING-8

R. Tanaka, T. Fukui, Y. Furukawa, T. Hatsui, T. Hirono, N. Hosoda, M. Ishii, M. Kago, A. Kiyomichi, H. Maesaka, T.Masuda, T. Matsumoto, T.Matsushita, T. Ohata, T. Ohshima, T. Otake, Y. Otake, C. Saji, T. Sugimoto, H. Takebe, M. Yamaga, A. Yamashita, RIKEN-JASRI Joint Project for SPring-8 XFEL, Hyogo, Japan

Abstract

The construction of the X-ray free electron laser facility (SACLA) in SPring-8 has started in 2006. After 5 years of construction, the facility has completed to accelerate electron beams in February 2011. The C-band linac accelerates beams up to 8GeV. The beam size is compressed to a length of 30fs, and the beams are introduced to the insertion devices to generate 0.06nm X-ray laser. The first SASE X-ray laser was observed at the beginning of June 2011. The first scientific experiment will start at the end of October this year. The control system for SACLA adopts the 3-tier standard model by using MADOCA framework developed in SPring-8. The upper control layer consists of Linux PCs for operator consoles, Sybase RDBMS for data logging and FC-based NAS for NFS. The lower consists of Solaris-operated VME systems and the PLC is used for slow control. The Device-net is adopted for the frontend devices to reduce signal cables. The beam-synchronized data-taking link is installed to meet 60Hz beam operation. The accelerator control has gateways to the facility utility system not only to monitor devices but also control the tuning points of the cooling water.

FACILITY STATUS

The X-ray Free Electron Laser (XFEL) is generated by a combination of a high-energy linear accelerator and a set of insertion devices (ID). The XFEL project in SPring-8, now called SACLA, has started in 2006 with the success of SCSS prototype accelerator [1]. The SACLA facility is designed to be compact to fit within 700m available spaces in the SPring-8 site with lower construction cost.

Figure 1: A bird eyes view of the SACLA facility. The electron beams are generated and accelerated from the right hand side to the left inside the linac yard. A picture of SPring-8 site is imposed.

The main component of the SACLA accelerator consists of 64 C-band RF units of 5.7GHz operation frequency to accelerate electron beams up to 8GeV. The initial beam length is compressed to produce the laser peak power generated by a length of 30fs electron beams. The 8GeV-electron beams are introduced to the 18 units of ID to generate less than 0.1nm X-ray laser. A total length of the accelerator is 400m, and that of ID is 230m. The SACLA has a unique experimental hall for co-operative experiments that uses both X-rays from SACLA and SPring-8 for example pump-up and probe experiments. The SACLA facility is shown in Figure 1.

The facility construction has completed in February 2011. The accelerator structure conditioning with RF high-power started from several months before the facility completion. The electron beam commissioning has started right after the completion. The electron beams were successfully accelerated up to 8GeV and the first light from an alignment ID was observed shortly after. After three months beam tuning, high brilliant SASE X-ray was observed at the beginning of June as can be seen in Figure 2. The SASE laser signal has homogeneous round shape, and the maximum laser power is ~4GW. The laser beam intensity is quite stable (~18%) to be ready for coherent X-ray experiments.

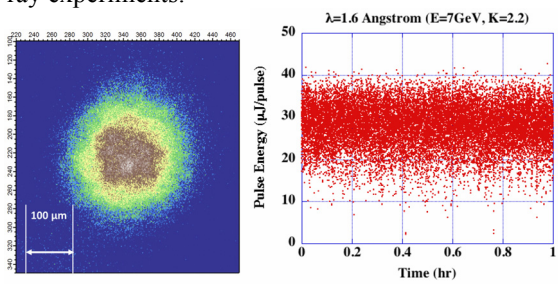


Figure 2: The observed profile of the SASE laser signal at 110m downstream of the last ID with 10keV (left). And the measured laser intensity stability (~18% fluctuation) during one-hour operation (right).

The laser wavelength has been improved from 0.16nm in June to 0.08nm at the end of July along with the progress of the beam tuning, as can be seen in Figure 3. Now, the laser beams are reproducible [2]. The electron beam tuning continues to achieve the laser coherency saturation and the first scientific experiment will start at the end of October this year.

The SACLA control system started smoothly at the early stage of the electron beam commissioning. The established MADOCA control framework helped smooth rise up of the SACLA control system and reduced time-consuming debugging work for the operation software.

Because of a large number of control points, however the control system had to be tuned suitably to meet high load, and system reliability was achieved along with the progress of the beam commissioning.

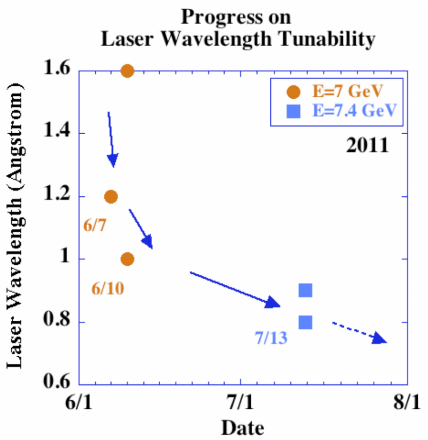


Figure 3: Improvement of laser wavelength of SACLA. The wavelength improved from 0.16nm to 0.06nm after two months beam tuning.

DESIGN CONCEPT

The design of the control system for SACLA is based on that of for SCSS test accelerator [3]. SCSS was constructed for the principle study of the free electron laser mechanism by using a linac and technology assurance towards the coming 8GeV XFEL machine, SACLA. We used common control devices for each subsystem of SACLA as much as possible [3]. This comes from the short construction period for the fast start up and less human resources coming from the joint project management sharing with SPring-8 operation.

Methodology Toward X-ray Laser

The basic elements for the SACLA control system shares commonly with those of SCSS, except for the difference coming from energy difference. The electron beam energy of SCSS is 250MeV on the other hand SACLA is 8GeV. This difference results the difference of the number of elements of the control system to build and equipment precision coming from the short wavelength. The wavelength of SCSS laser is around 50nm, on the other hand SACLA generates very short X-ray less than 0.1nm. The short wavelength also requires fine alignment of accelerator components and a series of ID, precise LLRF phase control, stable temperature both for high frequency timing signals and cooling water for accelerating structure [4]. Hence, we realized the facility utility control from the accelerator control [5].

Standard Control Structure with 3-tire Model

The SCALA control system follows the standard control structure with the established 3-tire model. The upper control layer consists of PCs running with SuSE Linux for operator consoles, Sybase RDBMS for data logging and FC-based NAS for NFS file server. The RDB data schema and the file server configurations are copied from existing SPring-8 control system with minimum modification. The lower layer consists of Solaris-operated VME systems for the faster control. The programmable logic controller (PLC) is used for slow control for example vacuum systems. The Device-net connected to the PLC module, is adopted for the frontend device control to reduce the number of signal cables. The VMEbus systems have a beam-synchronized data-taking link of a shared memory network to meet 60Hz beam operation for the beam tuning diagnostics [5]. The upper and lower layers are interconnected via many Gigabit Ethernet switches on which the MADOCA middleware runs. The 10GbE is introduced as a backbone data transfer line for user experiment. The out going 10GbE line is connected to the 10PFlops K-supercomputer in Kobe for the on-line analysis of experimental data that will be generated from two-dimensional X-ray detector. The schematic view of the control system is shown in Figure 4.

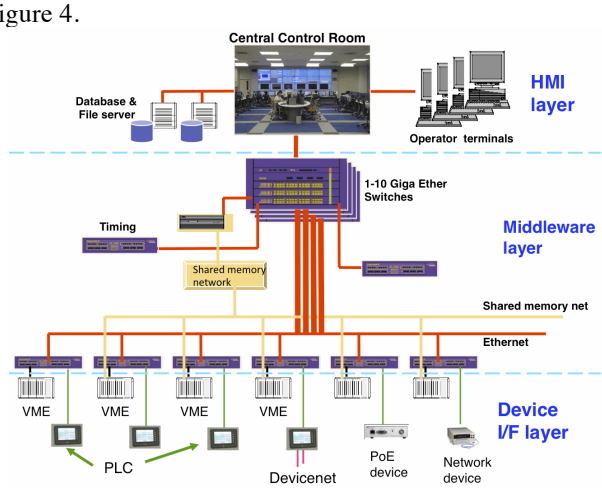


Figure 4: A 3-tier schematic structure of the SACLA control system. Beam commissioning is performed at the local control room nearby SACLA facility.

SACLA has an electron beam transport line to inject the electron beams to the SPring-8 storage ring as shown in Figure 1. At present, the injection is not scheduled yet, but SACLA works independently by using its own beamlines for X-ray FEL experiments. In order to avoid unexpected interference to the SPring-8 duty operation and guarantees independent operation between two facilities, the control network zone between SACLA and SPring-8 is loosely coupled. Also, the server computers such as a database machine and a NAS-based NFS file server are installed separately. On the other hand, the program development environment for SACLA shares with SPring-8 for the common software development. This configuration is good for the cooperative software development and timely program installation to the SACLA server without interference to the SPring-8 user-mode operation.

The number of equipment used for SACLA control is listed in the Table 1.

Table 1: The Number of Equipment for Controls

Equipment	SACLA	SACLA+SPring-8
Consoles	27	332
VME systems	175	450
Interlock stations	122	812
Network switches	181	758

EQUIPMENT CONTROL

The VMEbus systems and PLC stations are main base components of the SACLA accelerator control [4]. The C-band linear accelerator consists of mainly accelerating structures with 5.7GHz RF systems. We made one RF control unit and repeated the same configuration as much as needed. Additionally, fine timing system and beam monitoring system is necessary for the beam tune. Interlock systems are installed for the machine protection system (MPS) and personnel protection system (PPS), as shown in Figure 5.

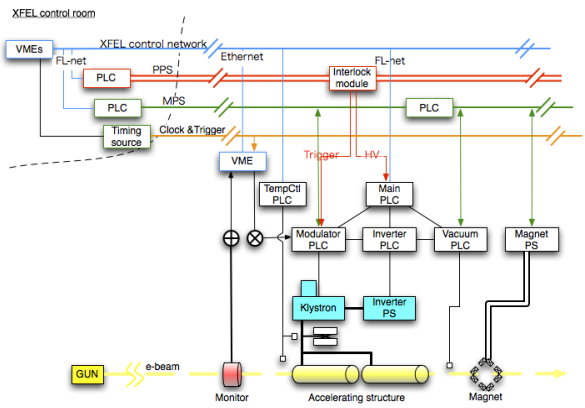


Figure 5: A schematic view of the SACLA control system. A typical control unit using PLC module for one C-band accelerating unit is shown in the figure.

The PLC system is used for RF high power (A, V) device control and the interlock systems. The PLC system is simple and suitable for the slow control with high robustness and reliability. On the other hand, the VMEbus system is suitable for the fast device control such as beam steering magnets and so on. To satisfy the stability requirements of the phase and amplitude of the cavity voltage, newly developed boards, such as DAC and ADC are equipped for LLRF VMEbus systems. A feedback control process (EMA on MADOCA) that runs on the VME CPU plays essential role to stabilize the phase and amplitude, which contributes the laser stability.

Implementation

The single-core CPU board, SVA041 already used in SPring-8, was selected for the SACLA control even though single core processor. The well-established CPU is good for smooth start-up of the control system. At the beam commissioning stage, the accelerator repetition rate is 1~ 10Hz, but it will go up to 60Hz and finally 120Hz (maybe 300Hz later). So far, signal process power of the Intel VMEbus CPU is enough but it is approaching to the marginal level, especially for the large data processing such as waveform data monitoring in on-line. A faster multi-core CPU with low power consumption is now available. The multi-core CPU (Core i7) will replace the current CPU if computing power is tight. The DMA and block transfer scheme was introduced to meet the waveform data handling from the 16bits ADC boards..

The FL-net is a FA-link that interconnects VMEbus systems and PLC stations. The FL-net link is divided into several links to keep independency of the device groups. The multiple links on the single VMEbus system is handled by using the virtual machine technology, Solaris10 Container. The I/O board configuration of the VMEbus had to adjust such that the I/O boards do not share one bus together with the large-data handling boards and also the slow control FL-net links. We prepare the monitoring and notify mechanism for FL-net unexpected link-down.

Database and Servers

The SPring-8 database system can process many data as we expected, however SACLA database system used almost full CPU power. The heavy load sometimes caused unexpected delayed responses to the accelerator operation GUI. Actually, the number of data was about factor two more than we estimated at the beginning of the database design phase. We adjusted the Sybase RDBMS configuration such as the procedure cache size, data cache size, the size of data log area and table lock mechanism. And we, also, optimized the periodic machine data-taking cycle to ease the database load. The alarm surveillance and monitoring programs handle many watching points as well. Especially at the alarm process start-up time, the heavy accesses from the alarm programs to the database deteriorated operation response.

We will add a 3GHz 12core FT-server with 48GB memory together with the current database machine of a 2GHz 8core CPU with 16GB (now 48GB) memory. We don't change NAS-based disk system because it has no problem. We, also, will tune the alarm software algorithm for much efficient surveillance and lighter database access.

Signals of Equipment

The number of equipment signals of SACLA is much larger than that of the SPring-8 accelerator complex, on the contrary to the initial estimation. The comparison of the data size is listed in Table 2.

Table 2: The Number of Equipment Signals

Accelerator	Analogue points	Digital points
SACLA	19,500	225,000
SPring-8 (Li,Sy,SR)	22,000	90,500

The estimation is the base to select the database computer specification because the CPU load depends on the size of the database schema, hence the size of the data,

however the NAS storage provides enough data transaction performance. The difference originates in the number of the units of RF acceleration structures in SACLA.

FACILITY CONTROL

The facility utility control of SACLA has unique features [5]. The accelerator beam tuning to achieve the stable laser imposes sever stability to the RF cooling water and electric power for example. The facility control has a gateway to the accelerator control to receive "put" command from the accelerator side. If the observed temperature deviated from the monitoring point, the water set point would be actively adjusted from the accelerator side. The precision of the monitoring sensors for the cooling water temperature was 0.01C, however the precision has to be improved to 0.001C resolution because the RF phase of the C-band shifted according to the water temperature drift.

The room temperature of the accelerator house and the spaces for the control racks of the LLRF systems have been monitored and kept into the database right after the completion of the facility building. The long accumulated temperature data about one year is useful to understand the stability and drift of the room temperature before the beam commissioning.

BEAMLINE CONTROL

Finally, SACLA will have five beamlines for the experiments. Now, one beamline (BL3) is ready for the scientific experiment. The application layer of the SACLA beamline control uses MADOCA identical to that of SPring-8 in order to achieve smooth migration of the current software developers from SPring-8 to SACLA. The beamline control uses the VMEbus system as well. The beamline components consist of slits, screen monitors, beam position monitors, a monochromator and so on, those have equipment interfaces similar to the SPring-8 beamlines. Additionally, the synchronized data-taking scheme is introduced to the beamline control [6]. In SACLA, this scheme takes the set value of optical devices and pulse motor positions, and sends these data to the beam-synchronized data-taking system for the accelerator. The data will be used for the experimental analysis together with the accelerator status. The beamline control ramped up smoothly and has been working with good stability.

DETECTOR DATA ACQUISITION

A linac-based FEL facility with the small number of beamlines has a feature such that a linear accelerator, insertion devices, an X-ray detector and data acquisition system (DAQ) for experiments have to work with required performance as a whole in order to generate the scientific results. In SACLA, the accelerator control framework, MADOCA, was introduced to the DAQ for the experimental data taking [7]. MADOCA interfaces between the accelerator control and the detector DAQ system to provide electron beam information to the laser experiment system, for example beam tag numbers as shown in Figure 6.

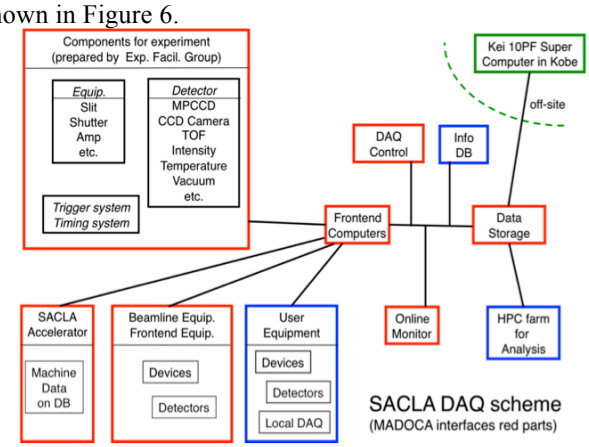


Figure 6: A schematic view of the SACLA DAQ for the X-ray detector. MADOCA plays as the middleware.

The multi-port CCD (MPCCD) sensor developed for SACLA experiments as workhorse detector has 0.5M pixels of 16bits data/pixel. Largest detector arrays consist of twelve MPCCD sensors. The DAQ system has to handle the data rate up to 720MB/s/system for 60Hz beam operation. The FPGA-based VME boards take the MPCCD image data via CameraLink interfaces. The data is compressed at the VME board then sent to the 240TB Infini-band storage system via data-handling PC servers. Whole the are pre-processed by a PC cluster, and transferred to the 10PFlops supercomputer "K computer" via 10Gbps Ethernet for on-line data mining and post analysis. The DAQ system is ready now and waiting for scientific experiments scheduled at the end of October this year.

REFERENCES

[1] T. Shintake et al., "Status of SPring-8 compact SASE source FEL project", NIM A21188, Nuclear Instrum. and Meth., In Phys. Resea. A507(2003)382-397.

[2] H. Tanaka, "Status Report on the Commissioning of the Japanese XFEL at SPring-8", in the proceedings of IPAC 2011, San Sebastian, Spain, 2011.

[3] T. Fukui et al., "STATUS OF THE SCSS CONTROL SYSTEM, ICALEPCS2005, Geneva, Switzerland, MO3.2-10 (2005)

[4] T. Fukui et al., "STATUS OF THE X-RAY FEL CONTROL SYSTEM AT SPRING-8, ICALEPCS2007, Knoxville, USA, TOAA03 (2007)

[5] T. Masuda et al., "FACILITY UTILITY CONTROL SYSTEM OF XFEL/SPRING-8", ICALEPCS2009, Kobe, Japan, TUP095 (2009)

[6] T. Ohata et al., "DESIGN OF AN XFEL BEAMLINE DAQ SYSTEM", ICALEPCS2009, Kobe, Japan, TUP095 (2009)

[7] M. Yamaga et al., "Event-Synchronized Data Acquisition System of 5Giga-bps Data Rate for User Experiment at the XFEL Facility, SACLA", in these proceedings of ICALEPCS2011, Grenoble, France

STATUS REPORT OF THE FERMI@Elettra CONTROL SYSTEM*

M. Lonza, A. Abrami, F. Asnicar, L. Battistello, A.I. Bogani, R. Borghes, V. Chenda, S. Cleva, A. Curri, M. De Marco, M. Dos Santos, G. Gaio, F. Giacuzzo, G. Kourousias G. Passos, R. Passuello, L. Pivetta, M. Prica, R. Pugliese, C. Scafuri, G. Scalamera, G. Strangolino, D. Vittor, L. Zambon, Sincrotrone Trieste, Trieste, Italy

Abstract

FERMI@Elettra is a new 4th-generation light source based on a seeded Free Electron Laser (FEL) presently under commissioning in Trieste, Italy. It is the first seeded FEL ever designed to produce fundamental output wavelength down to 4 nm with High Gain Harmonic Generation (HGHG). Unlike storage ring based synchrotron light sources that are well known machines, the commissioning of a new-concept FEL is a complex and time-consuming process consisting in thorough testing, understanding and optimization, in which a reliable and powerful control system is mandatory. In particular, integrated shot-by-shot beam manipulation capabilities and easy-to-use high level applications are crucial to allow an effective and smooth machine commissioning. This paper reports the status of the control system and the experience gained in two years of alternating construction and commissioning phases.

INTRODUCTION

FERMI@Elettra is a linac-based Free Electron Laser designed to supply photons in the soft X-ray spectral range. The 200 m long accelerator consists of a high brightness photo-cathode gun working at up to 50 Hz repetition rate, a 1.5 GeV normal conducting linac and two bunch compressors. The accelerated electron bunches, together with a tunable UV seed laser beam, are sent into a chain of undulators where they generate ultra short (<100 fs) and high peak power (~GW) coherent photon pulses with variable polarization. Two distinct undulator chains will be available, FEL-1 and FEL-2, covering the entire spectral range from 100 to 4 nm in the first harmonic.

Three beamlines have been designed so far, dedicated to different scientific areas: Low Density Matter (LDM), Elastic and Inelastic Scattering (EIS), and DIffraction and PROjection Imaging (DIPROI). The particular characteristics of the FEL photon beam enable time resolved experiments to study ultrafast dynamics and transient phenomena of matter under extreme irradiation conditions.

The FERMI@Elettra project has developed through several installation and commissioning periods and the achievement of intermediate milestones: first electrons from the photo-cathode gun in August 2009, linac energy at 1.2 GeV in September 2010, beam transported in the undulator hall in October 2010 and first free electron lasing in December 2010 [1].

Figure 1: Aerial view of the Elettra 2.4 GeV storage ring and the FERMI@Elettra Free Electron Laser.

OVERVIEW OF THE CONTROL SYSTEM

A complete description of the FERMI@Elettra control system is given in [2].

Fig. 2 shows the control system architecture. The low level computers are VME crates with PowerPC CPU boards or rack-mount Intel-based servers both running the GNU/Linux operating system with the Adeos/Xenomai real-time extension. Siemens S7 PLCs have been extensively employed in protection and safety systems, as well as for the low level control of undulators. In the control room, Linux PCs equipped with two or four monitors are used as operator workstations, while Intel-Xeon servers with a number of Xen virtual machines provide general common services such as network file system, remote boot, network services, database facilities, archiving system, etc.

The control system backbone is a pervasive Gigabit Ethernet network serving all of the machine tunnels and service areas of the facility. In addition, several isolated segments of data network are used to locally access Ethernet-enabled devices. The use of Ethernet as a fieldbus allowed us to reduce the number of low level computers (~70) necessary to interface a high number of devices.

A distributed real-time framework based on an in-house developed protocol called Network Reflective Memory (NRM) and a dedicated Ethernet network connecting all the low level computers, allows developing distributed

*This work was supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3.

real-time applications exchanging shot-by-shot data at the linac repetition frequency (10-50 Hz) [3].

The Tango toolkit has been adopted as control system software. Tango device servers are mostly written in C++ and Python, while graphical interfaces and control room applications are developed with QTango and Matlab. A high level software framework has been created to provide machine physicists with tools and methods to develop software applications for controlling the machine through its model.

The software development and deployment process relies on CVS. Bug reports and new requests are collected and managed using a ticketing system based on Bugzilla. Programmers develop the software using dedicated reference computers. The source code is then tagged on the repository, checked out and compiled on a clone of the field systems before being deployed in production. The system administrator hence performs the final installation keeping track of the details on an internal wiki page.

TWO YEARS OF MACHINE COMMISSIONING AND OPERATIONS: KEYS TO SUCCESS

The particular approach adopted in the project schedule, with alternating periods of installations and commissioning operations lasting in average five and nine weeks respectively, and the difficulties inherent in the commissioning of a novel machine, have set new challenges in the design and implementation of the control system. Since the beginning of each commissioning phase, physicists need to smoothly operate via the control system the machine components that have just been installed. Moreover, while performing the complex and time-consuming commissioning operations, the control system must effectively support them without inconvenience to their activities. For these reasons the control system must be flexible, high-performing and reliable: a service rather than a concern.

The experience gained in two years of installations and machine commissioning records several difficulties but emphasizes the importance of a few strategic factors that made it possible to meet the requirements mentioned above. They are in particular the standardization of hardware and software, the choice of Tango, the real-time framework and the use of Matlab.

Standardization

Special care has been taken from the beginning of the project in order to possibly adopt the same hardware/software technologies as well as the same control system architecture throughout the whole facility, from the photo injector to the experimental stations. Standardization involves several aspects of the control system: hardware interfaces (mostly Ethernet and serial), low level computers (VME/PPC or Intel servers, depending on the required performance and hardware interfaces), system software (Linux and Tango) and graphical interfaces (QTango and Matlab). Forcing standardization has been sometimes hard and slightly more expensive, but the benefits in reducing the development and maintenance costs are evident. A particular case is the adoption of the YAMS motor controller, an in-house design using commercial components, utilized for most of the stepper motors of the facility [4]. Being a flexible system it fits to any motion application while maintaining a common interface to the control system. More than 400 axes will be eventually controlled by YAMS.

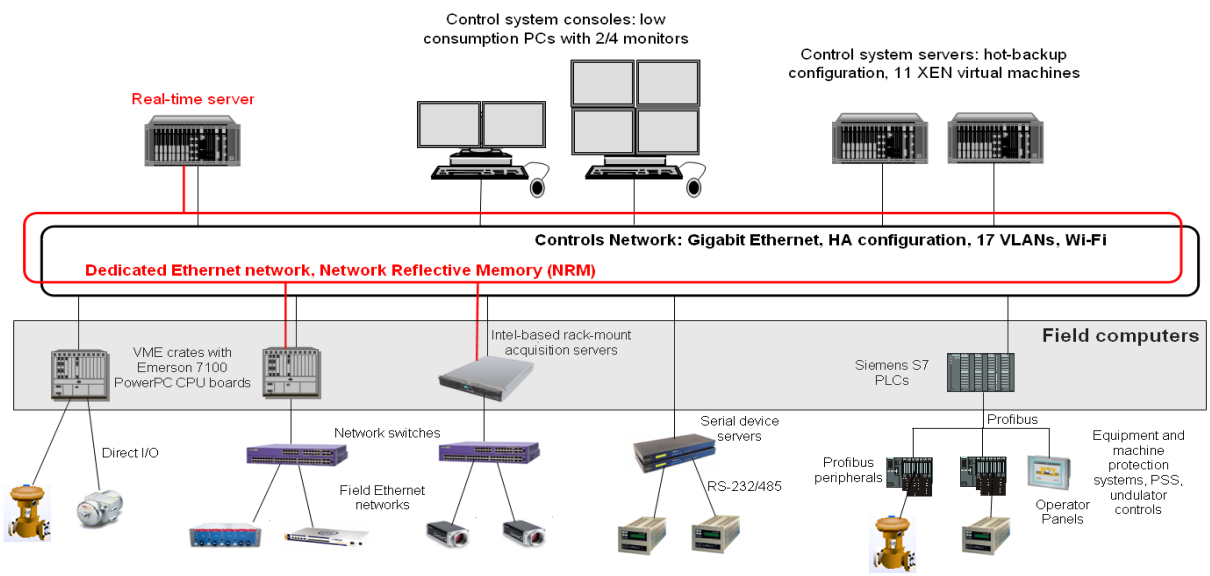


Figure 2: Architecture of the FERMI@Elettra control system.

Tango

The Tango toolkit is the real "glue" of the control system providing an effective integration of the controlled technical systems and the software controlling them. From the client point of view the machine components are seen through a uniform API interface regardless of the way they are connected to the control system.

The device model, which is the fundament of Tango, offers an object oriented approach to controls issues, with important benefits for the software development and maintenance. Each equipment or physical device connected to the control system is assigned a Tango device which exports an API with attributes and commands. Further hierarchical levels of Tango devices with a higher level of abstraction can then be developed to aggregate several devices, add more sophisticated functionalities or offer different views of the same device to the client level. This possibility has been widely utilized in FERMI@Elettra; following a general rule that we adopted, all of the coordination, processing and automation functionalities have been implemented in the device server level, leaving to the GUIs only display duties.

Tango is behaving as required in terms of reliability and performance, and confirmed to be mature and stable.

Real-time ramework

FERMI@Elettra is a pulsed machine with repetition rate up to 50 Hz. A distributed real-time framework well integrated into the control system provides the capability to measure and manipulate the laser, electron and photon beams, as well as to close synchronized feedback loops on a pulse-to-pulse basis.

All the relevant monitor points (e.g. electron and photon diagnostics) and control points (e.g. power supplies) are synchronized with the bunch trigger. Moreover, a real-time time-stamp called "bunch number" is distributed to all of the low level computers via the NRM.

The diagnostic electronics provide shot-to-shot data to the low level computers through real-time interfaces; data are tagged with the bunch number and stored in local circular buffers that can be easily read by high-level applications through specific Tango servers. The same data can also be placed in the NRM and shared in real-time with the other control system computers (see Fig. 3).

Most of the power supplies allow synchronized current settings. The values can be generated, sent to the low level computers through the NRM and set on the power supply shot by shot in real-time. Alternatively, an arbitrary waveform can be loaded into a local buffer through a Tango server and then used to set the power supply shot by shot after a trigger command.

The above described capabilities have been widely used during the machine commissioning. Relevant examples are FEL jitter studies carried out by correlating shot-to-shot data provided by photon and electron diagnostics, and fast emittance measurements using synchronized quadrupole power supplies and fluorescent screen CCDs.

A centralized real-time server can run synchronized feedback loops by reading sensors and setting actuators through the NRM at the linac repetition frequency. For now a fast trajectory feedback has been developed to correct the electron beam position in various locations of the machine [5].

Matlab

Thanks to the availability of effective Tango bindings, Matlab has been chosen to easily develop scripting procedures and GUIs interacting with the machine through the control system. Matlab has also been interfaced to simulation codes like Elegant [6], thus opening the way to the physicists for the development of their own machine physics applications and measurement procedures employed during the accelerator commissioning.

Matlab GUIs are also useful for graphical panels prototyping carried out by non-professional programmers such as physicists or specialists of technical systems. The GUIs are developed, used and modified several times, and eventually translated to QTango by controls people once they are stable.

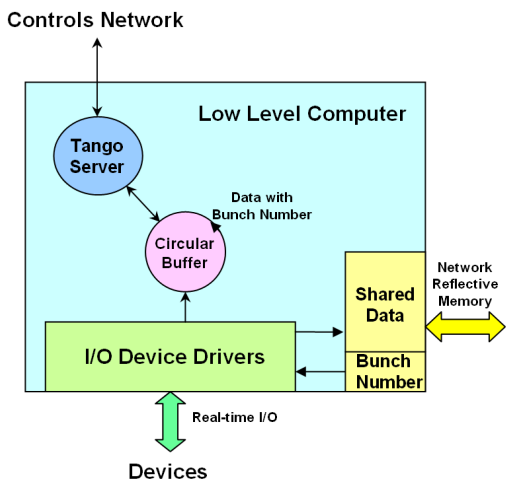


Figure 3: Software architecture of a low level computer acquiring real-time data.

EXPERIMENTAL STATIONS CONTROL AND DATA ACQUISITION SYSTEMS

Experiment controls are an extension of the machine control system and employ the same technology and architecture [7]. For practical reasons the control system of the experimental stations is logically separated from the machine: each of the stations has its own Tango database running on a dedicated virtual machine and separated VLANs in the controls data network. Access to the machine control system from the experimental stations is realized by providing a restricted view of the

machine devices through special Tango servers acting as bridges.

Data acquisition and online processing applications are developed using Tango on Linux machines. High-level applications can be developed both on Linux and Windows platforms using C/C++, Python, LabView, IDL or Matlab.

The NRM provides the experimental stations with the bunch number and pulse-to-pulse data from the upstream electron and photon diagnostics. This allows associating to the data acquired in each shot from the experiment detectors the characteristics of the corresponding FEL photon pulse, such as position, intensity, wavelength and bandwidth.

A scientific data storage system featuring a capacity of several hundreds of terabytes is being designed to accommodate the considerable amount of data that will be generated by the detectors with sustained data rate up to 400 MB/s. Dedicated acquisition servers will receive data from the detectors through high-speed real-time interfaces and will directly send them via dedicated 10 Gb/s Ethernet links to an online processing server, which will reduce data and eventually transfer them to a centralized storage system.

CONCLUSION AND OUTLOOK

After the commissioning of the machine and optimization of the FEL process in the first line of undulators (FEL-1), in July 2011 a coherent photon beam having the expected intensity and stability in a tunable wavelength range from 52 to 20 nm has been obtained, allowing the first rudimental experiments in the beamlines. In the next months, further optimizations will be carried out after the commissioning of the second bunch compressor and of the X-band RF cavity devoted to the linearization of the longitudinal phase space. In 2012 FEL-2 will be commissioned and more time will be gradually dedicated to the experiments, accomplishing the transition of the facility from commissioning to users operation.

The control system will evolve according to the new requirements. In particular, a number of additional feedback loops will be provided to stabilize the main beam parameters, such as charge, energy, bunch length and time of arrival. Moreover, further high-level applications will be developed to let the operators run the machine effectively and autonomously during users shifts.

REFERENCES

[1] S. Di Mitri, "Commissioning and Initial Operation of FERMI@Elettra", IPAC 2011, San Sebastián, September 2011.

[2] M. Lonza et al., "The Control System of the FERMI@Elettra Free Electron Laser", ICALEPCS2009, Kobe, October 2009.

[3] L. Pivetta et al., "The FERMI@Elettra Distributed Real-time Framework", these proceedings.

[4] A. Abrami et al., "YAMS: a Stepper Motor Controller for the FERMI@Elettra Free Electron Laser", these proceedings.

[5] G. Gaio et al., "Commissioning of the FERMI@Elettra Fast Trajectory Feedback", these proceedings.

[6] C. Scafuri, “The FERMI@Elettra Online Modelling Toolkit”, ICALEPCS2009, Kobe, October 2009.

[7] R. Borghes et al., "Control and Data Acquisition Systems for the FERMI@Elettra Experimental Stations", these proceedings.

CONTROL SYSTEM IN SWISSFEL INJECTOR TEST FACILITY

M. Dach, D. Anicic, D. Armstrong, K. Bitterli, H. Brands, P. Chevtsov, F. Haemmerli, M. Heiniger, C. E. Higgs, W. Hugentobler, G. Janser, G. Jud, B. Kalantari, R. Kapeller, T. Korhonen, R. A. Krempaska, M. Laznovsky, T. Pal, W. Portmann, D. Vermeulen, E. Zimoch,
Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

The Free Electron Laser (SwissFEL) Injector Test Facility is an important milestone for realization of a new SwissFEL facility at Paul Scherrer Institute (PSI) in Villigen, Switzerland. The first beam in the Test Facility was produced on the 24th of August 2010 which inaugurated the Injector operation. Since then, the beam quality in various aspects has been greatly improved.

The primary goal of the Injector Test Facility is to demonstrate a high-brightness electron beam which will be required to drive the SwissFEL main linac. The Injector further serves as a platform for the development and validation of accelerator components needed for the SwissFEL project.

This paper presents the current status of the Test Facility and is focused on the control system related issues which led to the successful commissioning. In addition, the technical challenges and opportunities in view of the future SwissFEL facility are discussed.

INTRODUCTION

The SwissFEL project at the Paul Scherrer Institute (PSI) foresees the realization of a SASE X-ray Free Electron Laser (FEL) operating at 0.1–7 nm photon wavelength by 2016 [1]. To minimize the facility length, and thus cost, the concept aims at relatively low electron beam energy. For the extensive study of the generation, transport and compression of high brightness electron beams and for developing the necessary components, PSI is commissioning the SwissFEL Injector Test Facility, a highly flexible 250 MeV linear electron accelerator [2].

The Injector Test Facility consists of a laser driven RF gun followed by an S-band booster section, a bunch compression section and a diagnostics section featuring an RF deflector and a series of FODO cells for emittance measurements [1].

The commissioning of the test facility proceeds in three phases: in a first step (phase 1), the gun section with some dedicated diagnostics was put into operation [3].

The second phase, lasting from August 2010 to May 2011 referred to the assembly of the full accelerator, with the exception of the bunch compression chicane, which was installed in July 2011.

The bunch compressor chicane will be put into operation during the third commissioning phase, starting later in 2011.

CONTROL SYSTEM OVERVIEW

The control and data acquisition system in the Injector Test Facility is based on a concept derived from the Swiss Light Source (SLS), in order to reduce the implementation effort due to limited man-power resources. Nevertheless, far more effort than originally foreseen was required to develop new applications and to elaborate on an infrastructure that fits well to the different characteristics of the FEL (i.e. linear pulsed machine) as opposed to the SLS (i.e. storage ring). Ultimately, an integrated system based on the EPICS toolkit was provided for both injector control and experimental data-acquisition.

EQUIPMENT INTERFACE

Equipment to be controlled and monitored connects to the control system mainly via VME crates (which fulfil VME64x standard). This concept was directly derived from the SLS [4]. An approximate total of 40 VME crates have been installed in the Injector Test Facility. In addition, non-VME solutions were elaborated upon to meet FEL requirements and to make use of new technologies.

An image capturing and acquisition system based on the Firewrire camera has been provided. Around 40 cameras were installed mainly for diagnostic screen monitor systems. Such cameras are interfaced to EPICS via the PC/Linux servers. The serial devices are controlled by means of the custom made embedded controllers based on the Virtex-4 FPGA. Such systems run EPICS on top of Linux.

Another embedded solution based on Virtex-4 FPGA is the “data concentrator” controller which is used to control magnet power supplies via fibre-optical links.

Temperature and humidity are measured by means of dedicated sensors interfaced to EPICS via the MSCB bus with embedded controllers.

NETWORK

The network of the Injector Test Facility was designed in view of the requirements for SwissFEL. We assumed that the number of network ports needed will be too high to fit into one class C subnet. Figure 1 schematically shows the layout of the network. Every circle in the diagram represents a class C network with 254 addresses. The connections of the different subnets of the EPICS network are controlled by the so called Channel Access Gateways, computers running software to minimize network traffic. The control room is the central location from where programs, mostly for commissioning, are run.

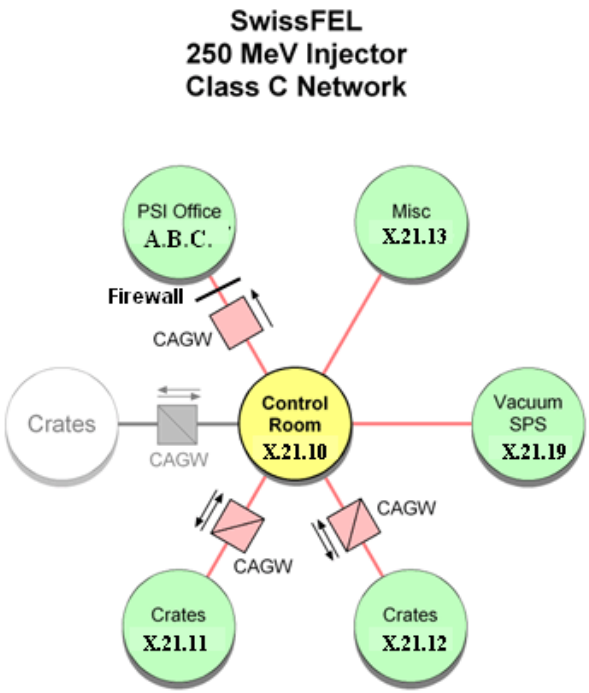


Figure 1: A schematic layout of the Injector Test Facility network consisting of several class C subnets.

SOFTWARE INSTALLATION

The control system software and configuration data is installed in a separate directory residing in a dedicated NFS file server. All files required to run VME and Linux IOCs and all client application configurations are located here. The 'swit' standard tool is used to install all control system applications at PSI.

PHYSICS APPLICATION

The Injector Test Facility is the first pulsed electron linac at PSI, and as such, the controls group had no previous experience in writing the necessary high level physics applications. These were mainly supplied by the physicists and engineers responsible for the commissioning. The controls group provides support by maintaining the MATLAB Channel Access (mca) interface to EPICS. Certain applications requiring primitive data types not supported by mca have made the transition to the more robust mocha (MATLAB Objects for Channel Access) package, which uses an in-house, C++ channel access library (CAFE) [5].

A number of physics process variables (PVs) have been implemented as soft EPICS channels (i.e. channels not connected to any hardware device) that hold information relevant for physics studies. One example is the emittance measured at a certain position along the beam axis. The measurement is time-consuming and invasive, but its value remains of interest for some time afterwards. It is therefore made available to other applications and status displays through a dedicated EPICS PV..

OPERATOR APPLICATIONS

As a backbone for operator applications we use programs from the EPICS toolbox like medm, alarm handler, and archiver. They are mainly chosen for compatibility to SLS as all accelerators are operated from the same control room and by the same operators.

Using a style guide for the application developers supports a consistent behaviour of all applications and a common integration of control room applications into EPICS. It provides rules for the application developer and simplifies the usage of the applications for the accelerator operators.

TIMING

Timing has a crucial role in the injector facility. It not only provides the triggering mechanism but also supports synchronous and reliable transmission of machine critical data. The timing system is based on the global event distribution by Micro Research Finland. We use the latest generation of the event system which supports real-time data transfer parallel to the event distribution. Non-VME systems e.g. Linux-based PCs which require timing, e.g. Firewire camera servers, receive timing information by PCI Event Receiver cards. The timing system is well integrated into the control system through which all its function are accessible. Sequencer of the event generator requires programming at high rate because some subsystems of the machine require trigger rates different than the machine nominal rate (i.e. 10 Hz). The sequencer is therefore programmed at the highest required rate which is 100 Hz in our case. Timing system capabilities has enabled us to realize a beam-synchronous data acquisition system for on-demand pulse-to-pulse correlated studies of the machine behaviour

DIAGNOSTICS

In order to support the extremely challenging SwissFEL Injector beam parameters, electron beam diagnostics components need to provide high performance with excellent reliability. At the same time, they have to be flexible enough to incorporate any improvements, which may be necessary to validate the final accelerator performance. Most of the beam diagnostics elements are based on the designs successfully implemented at the SLS as well as at other FEL facilities.

The major SwissFEL injector beam diagnostics tools include: integrated current transformer, wall current monitor, Transverse Deflecting Structures (TDS), beam profile monitors [6], beam position monitors (BPM), Electro Optical bunch length Monitors (EOM), and Bunch Arrival time Monitors (BAM).

Electronics of BPM systems (button, stripline, and cavity) is implemented in VME64x, which is the PSI standard for real time applications [7]. The digitized signals from BPM pickups are processed by dedicated FPGA modules. Calculated beam positions, at different averaging rates, are pushed into the EPICS database and immediately become available for the PSI control system.

The BPMs provide a high resolution (up to 10 μm) for a large range of electron bunch charges (10-200 pC).

Optical screen monitors together with direct (synchrotron and diffraction) radiation monitors are used to visualize the transverse electron beam profiles. The main controls components of these monitors are video cameras and VME based stepper-motor driving hardware.

Non-invasive monitoring of the longitudinal charge distribution in electron bunches is provided by EOM systems. Such systems for the SwissFEL are designed and built at the PSI [8]. Their resolution (~100 fs) is adequate for the expected bunch lengths. Core EOM elements are an advanced fiber laser system with pulse generating, phase locking, and synchronization electronics talking to the external world via serial (RS232) and GPIB communication lines. All EOM components are fully integrated into the PSI controls, which significantly simplifies their operations.

BAM systems will allow one to determine the beam arrival time with the resolution of ~10 fs. BAM signals can be used, for example, in feedback loops preventing beam energy drifts and stabilizing RF amplitudes of the injector linac structures. Key BAM elements are remotely handled by VME stepper-motor drivers as well as standing alone high precision motor controllers and signal synchronization devices talking via serial (RS232) communication lines. The work on the BAM developments and their integration into EPICS is in progress.

LASER

Two laser systems are used during the commissioning of the injector test facility: a compact, turn-key Nd:YLF amplifier suitable for basic commissioning tasks and a more sophisticated Ti:Sapphire amplifier allowing longitudinal pulse shaping and wavelength tuning for detailed emittance studies [1]. Both of the lasers require many devises to be interfaced to the control system.

More than 50 stepper motors are used to control mirrors on the optical path from the laser to the RF gun. The end user is able to monitor the beam shape at various locations by means of several firewire cameras. The beam laser energy measurement is based on the photodiode readouts. The photodiode signals are first pre-amplified, integrated in the boxcar integrator and digitised by ADC. For the proper operation the laser based system requires stable conditions in terms of the temperature, humidity and air pressure. There are several sensors installed to monitor such parameters in the air-conditioned laser hutch. They are interfaced to the control system by means of the MSCB distributed monitors/controller nodes. One of the challenges from the controls point of view was to improve the beam pointing stability at run-time. This was achieved by a dedicated application which performs a beam spot analysis, and through the PID loop, controls the motor mirror along both the X and Y axis [9].

RF

There are in total 7 RF systems foreseen: an RF gun, four accelerating structures, one x-band cavity and a deflecting cavity. Each of these systems consists of three parts.

A modulator supplied with an integrated control system that has been interfaced to EPICS with a serial TCP/IP connection.

An interlock system and low level RF (LLRF) were developed at PSI, based on VME, and are fully integrated into EPICS.

ALIGNMENTS

Computer monitoring and movement control of several parts of the Injector Test Facility system are required.

A Gun Solenoid magnet mover system with five degrees of freedom (DOF), which is based on the SLS HEXGIR systems, has been installed.

A second system is currently under construction to allow controlled movement of the RF XBand diagnostic equipment at the end stage. The HEXGIR girder mover system is flexible and can be adapted to different mechanical girder layouts. It is highly configurable with all physical parameters defined in one file. The system was designed to allow six degrees of freedom movement if needed [10]. The bunch compressor dipole magnet system has a simpler alignment system for horizontal movement only. All of these systems use EPICS in a single VME crate.

CONCLUSIONS

The SwissFEL Injector Test Facility provided us with an opportunity to develop and evaluate new ideas in light of the future SwissFEL project. The successful commissioning (phase 1 and 2) revealed some points for future improvement. The stability and performance of the Channel Access Gateways is questionable. The choice of firewire camera standard seems to be far from optimal for highly distributed SwissFEL environment. The GigE standard was chosen to improve the stability and reliability of the existing video system in the future. The fast feedback system is under investigation.

REFERENCES

[1] T. Schietinger et al., "Commissioning Status of the SwissFEL Injector Test Facility", IPAC 2011, San Sebastian, Spain.

[2] M. Pedrozzi., "250 MeV Injector Facility for the SwissFEL Project", FEL 2009, Liverpool, UK; M. Pedrozzi (ed.), "250 MeV Injector Conceptual Design Report", PSI Report 10-05 (2010).

[3] T. Schietinger et al., "First Commissioning Experience at the SwissFEL Test Injector Facility" LINAC 2010, Tsukuba, Japan.

[4] S. Hunt et al., "Status of the SLS Control System", ICALEPCS 1999, Trieste, Italy.

[5] J. Chrin and M.C. Sloan, “CAFÉ, A Modern C++ Interface to the Channel Access Client Library”, These Proceedings, ICALEPCS 2011, Grenoble, France.
[6] M. Pedrozzi et al. “250 MeV Injector Facility for the SwissFEL Project”, FEL 2009, Liverpool, UK, 2009
[7] B. Keil et al. “The European XFEL Beam Position Monitor System”, IPAC 2010, Tsukuba, Japan, 2010.
[8] F. Mueller et al. “Electro Optical Sampling of Coherent Synchrotron Radiation for Picosecond Electron Bunches with Few pC Charge”, Beam Instrumentation Workshop 2010, Santa Fe, USA, 2010.
[9] T. Pal et al.,“Laser Pointing Feedback Control at the SwissFEL Test Injector”, These Proceedings, ICALEPCS 2011, Grenoble, France.
[10] A. Streun "Six-dimensional Girder Positioning", http://controls.web.psi.ch/TWiki4.1.2/pub/Main/HEXGIR/hexgir.pdf

STATUS OF THE CONTROL SYSTEM FOR THE EUROPEAN XFEL

K. Rehlich, DESY , Hamburg, Germany

Abstract

DESY is currently building a new 3.4 km-long X-ray free electron laser facility. Commissioning is planned for 2014. The facility will deliver ultra short light pulses with a peak power up to 100 GW and a wavelength down to 0.1 nm. About 200 distributed electronic crates will be used to control the facility. A major fraction of the controls will be installed inside the accelerator tunnel. MicroTCA was chosen as an adequate standard with state-of-the-art connectivity and performance including remote management. The XFEL [1] will produce up to 27000 bunches per second. Data acquisition and controls have to provide bunch-synchronous operation within the whole distributed system. Feedbacks implemented in FPGAs and as service tier processes will provide the required stability and automation of the FEL. This paper describes the progress in the development of the new hardware as well as the software architecture. Parts of the control system are currently implemented in the much smaller FLASH FEL facility.

INTRODUCTION

The front-end electronics will be distributed along the 3 km long machine. Since a major part of the XFEL is running below populated public areas, the electronics has to be placed inside the tunnel close to the accelerator components. This requires appropriate shielding against radiation and remote management capability of the hardware. Furthermore, shots will have repetition rate of 10 Hz and a length of a ms. The bunches are spaced with 1/4.5 MHz and 27000 bunches per second have to be recorded and processed. A timing system is planned to synchronize all front-end systems. Data of all subsystems will be transmitted to central services and have to be sorted to sync all bunches. A fast DAQ system, designed and operated for FLASH [2], will provide a common solution for multiple services. As common hardware platform MicroTCA has been defined as the standard for XFEL fast controls.

MICROTCA STANDARD

Hundreds of megabytes per second have to be transferred within a crate and to the central services. MicroTCA provides these fast transfer rates and offers GigaLinks as point-to-point connections between modules for e.g. fast feedbacks. And MicroTCA has a very well defined interface to manage crates and individual modules. This is described in the next chapter.

But, the original specification was driven by telco applications. Analog IO and the distribution of timing information were not foreseen. Therefore a group of laboratories together with industrial partners initiated a working group within the PICMG ® [3] to specify rear transition modules, clock and trigger distribution. One result of this committee is the MTCA.4 specification, which is available since October 2011. It is fully compatible to the older MicroTCA specification. All existing modules can run in this new extended standard. During the specification period institutes and industry have designed first prototypes.

The driving idea of this MTCA.4 standard is to build only a few complex front modules as AMCs (Advanced Mezzanine Cards) and many simple RTMs (Rear Transition Modules) to adapt the AMCs to different applications. AMC and RTM are connected by two shielded 30 pair ADF connectors. They support fast GigaLinks as well as sensitive analog signal transmission. Six pairs of one plug are defined as management and power interface of the rear module. RTMs are fully integrated in the crate management. Hot-swap as well as a protection against wrong inserted modules are defined.

In September 2011 DESY has successfully demonstrated a MicroTCA system controlling one super-conducting RF module, with 8 cavities, driven by one 5 MW klystron. The Low Level RF system (LLRF) was completely installed in a MicroTCA crate according to the MTCA.4 standard.

The setup was (Figure 2):

- Three 125 MSPS, 16 bit ADCs with 1.3 GHz down-converters as RTMs.
- One controller with a vector modulator to drive the klystron.
- One timing module to synchronize the LLRF with the linac.
- A CPU and disk.
- An MCH as a switch for Ethernet and PCIe and to manage the crate.

It was shown that MTCA.4 can operate sensitive analog

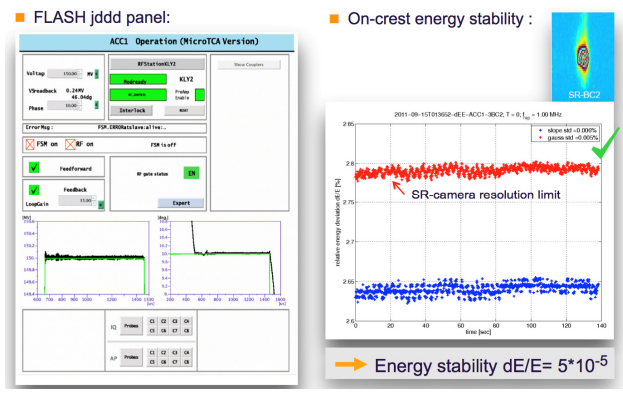


Figure 1: 800µs flat-top operation of FLASH with the new µTCA system.

electronics together with high frequency switched digital CPUs and FPGAs without degradation of signals quality.

The measured energy stability of the FLASH electron beam during 800 µs was better than $5*10^{-5}$ (Figure 1). This fulfills the XFEL requirements.

This demonstration included the full chain of new developed firmware, Linux drivers, DOOCS device servers and hardware management. In other words, the MicroTCA crate was completely integrated into the control system of FLASH. The device servers were written in a way that existing client programs find many names of the old VME system and required only minor modification to address new features.

Figure 2: The rack with the blue MicroTCA crate controls 8 superconducting cavities of one cryo module in FLASH.

HARDWARE MANAGEMENT

The hardware management operates on two levels. IPMI commands are available on the MCH Ethernet port for remote access. And within a crate PCIe is used to handle hardware events.

MicroTCA specifies IMPI functions for all hardware modules in a crate. It includes all AMC modules, fans, and power supplies. A DOOCS server, as a middle layer service, talks to all crates and dynamically detects inserted modules and monitors them. If a user inserts an AMC module the server creates a device instance for the module. It allocates data archives for the voltage and temperature readings and creates the corresponding addresses to be accessible for clients (Figure 4). During configuration of the server it is just required to specify the crate IP address, the other information is read from the hardware via well-defined IPMI commands. Furthermore, rebooting of a CPU with a frozen operating system can be initiated by power cycling an AMC via the IPMI DOOCS server.

Within a crate the PCIe signaling allows to handle removal and insertion of modules. In a running system one can remove e.g. one ADC board. The DOOCS server continues with the other ADCs and marks the removed one as faulty. After reinsertion of the ADC the server restores the registers and continues readout. This hot-swap does not require any operator actions. Linux drivers and DOOCS server processes automatically handle it.

MIDDLE LAYER SERVICES

As mentioned, XFEL will produce up to 27000 bunches per second. A data acquisition system (DAQ) has been implemented at FLASH to cope with the high data rates and to synchronize the data from a distributed system in one or more middle layer servers. The system is used for online processing of bunch related data, to archive all bunches of all monitors, and to store data of user experiments of the photon beam lines.

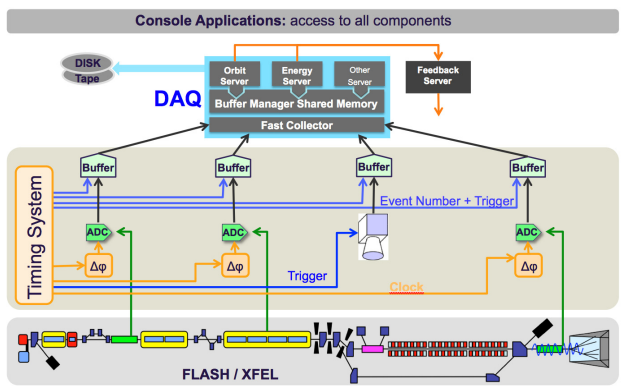


Figure 3: The timing system synchronizes the distributed controls. Data is collected in the central DAQ system.

The DAQ provides data of all BPMs for instance to calculate an orbit. This service was recently used to attach a slow orbit feedback to stabilize the beam in all sections of the accelerator (Figure 3). As a display and configuration tool jddd was used.

These three software packages, the DAQ, the feedback process and the jddd panel editor [4] are components to be used for XFEL commissioning. FLASH is used to test

the new hardware and software. Both, FLASH and XFEL will profit from this parallel development.

CONSOLE APPLICATIONS

The graphical operator panel design tool jddd is a further development for both machines. The goal of the design is to create a tool that is simple and can be used without knowledge of programming languages. Operators and subsystem experts are able to create themselves panels and can run them on multiple platforms. Jddd is Java based and the panels are stored in a Subversion repository. The panel can run on Mac OS, Windows and Linux operating systems.

Although jddd is usable without writing code, it has many complex widgets to construct highly dynamic panels. For example displays showing all running processes of the control system were created. During run-time these panels are dynamically updated by reading from a name server and device servers. See [4] for more information.

CONCLUSIONS

MicroTCA has been chosen as the standard platform for the XFEL. A first implementation of the LLRF controls based on this new standard has demonstrated that the most complex subsystem could be realized with excellent performance. This is a proof of principle for the new MTCA.4 standard. All required hardware and software components could be implemented and successfully tested. It includes hardware design, firmware, module management, Linux drivers, control system device servers and high level management server and display.

ACKNOWLEDGEMENTS

This work was only possible with the excellent contributions of many members of collaboration partners from Poland and Sweden, industrial partners, and the DESY groups MSK and MCS4 – many thanks to all of them.

REFERENCES

[1] http://xfel.desy.de
[2] http://flash.desy.de
[3] PICMG, http://www.picmg.org
[4] http://jddd.desy.de

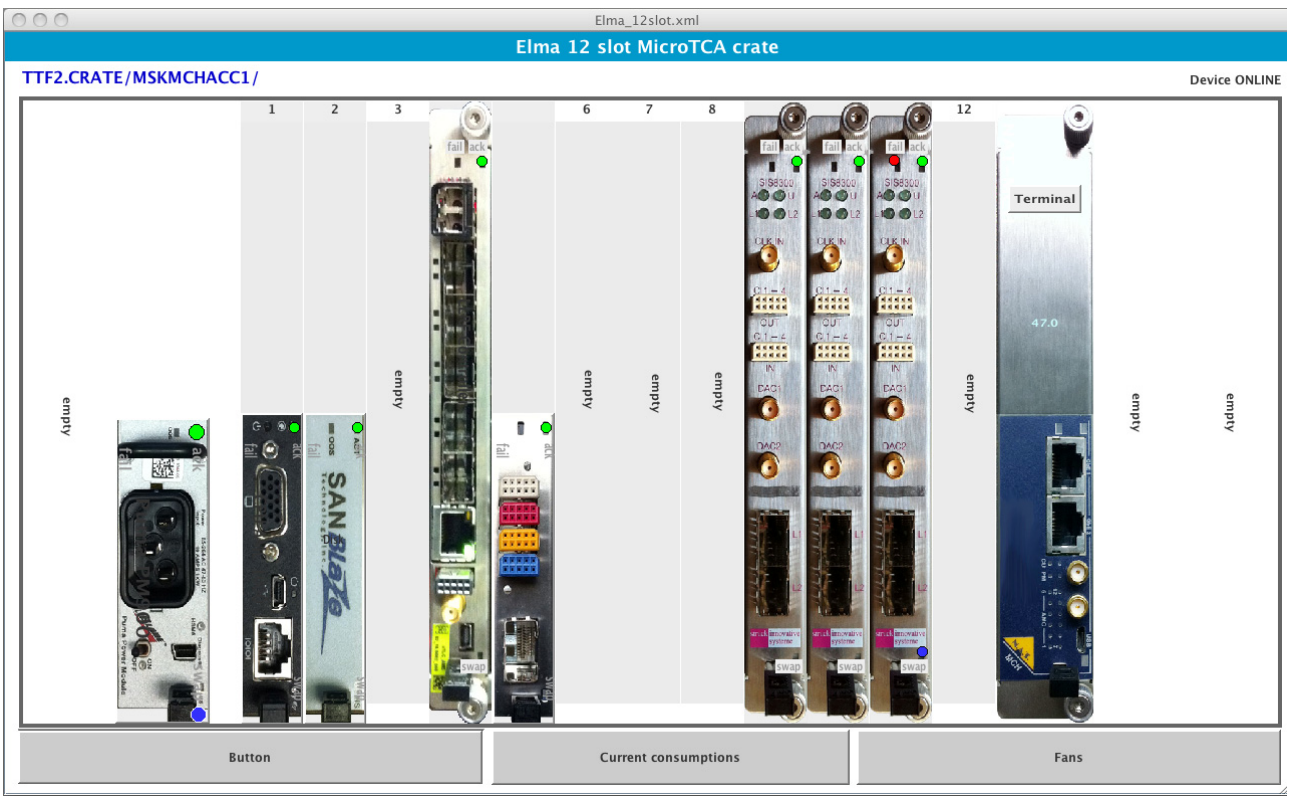


Figure 4: The MicroTCA crate displayed by jddd. A click on a module starts a panel with further details.

THE LASER MEGAJOULE FACILITY: CONTROL SYSTEM STATUS REPORT

J.P. Arnoul, F. Signol CEA/CESTA, Le Barp, 33114, France
J. Nicoloso, CEA/DIF, Bruyères le Châtel, 91297, Arpajon, France

Abstract

The French Commissariat à l'Énergie Atomique (CEA) is currently building the Laser MegaJoule (LMJ)[1], an up to 240-beam laser facility, at the CEA Laboratory CESTA near Bordeaux. It is a Megajoule class facility, designed to deliver a high energy to targets for high energy density physics experiments, including fusion experiments. LMJ technological choices were validated with the LIL, a scale 1 prototype of one LMJ bundle. The construction of the LMJ building itself is now achieved and the assembly of laser components is on-going.

The presentation gives an overview of the general control system architecture, and focuses on the hardware platform being installed on the LMJ, in the aim of hosting the different software applications for system supervisory and subsystem controls[2] (Fig. 4).

This platform is based on the use of virtualization techniques that were used to develop a high availability optimized hardware platform, with a high operating flexibility, including power consumption and cooling considerations. This platform is spread over 2 sites, the LMJ itself of course, but also on the software integration platform built outside the LMJ to provide system integration of various software control system components of the LMJ.

LMJ FACILITY

The Laser Megajoule facility (LMJ) is presently under construction at the CEA/CESTA site near Bordeaux (France). LMJ is an up to 240-beam laser system designed to study inertial confinement fusion (ICF) and the physics of extreme energy densities and pressures. LMJ will be capable of focusing its energy of ultraviolet light on to an extremely small micro-target in an extremely short space of time. The characteristics of this facility were defined to obtain the temperature and pressure conditions required to reach thermonuclear ignition. Experiments will involve tenths of a milligram of matter in which nuclear fusion reactions will be produced. LMJ is an element of the Simulation Program that aims the guarantee of the safety and reliability of French nuclear weapons. It is comparable to the US NIF facility. LMJ will welcome national and international scientific collaborations, in order to become a privileged place of scientific exchanges.

The LMJ building covers a total area of 40,000 m2 (300 m long x 150 m wide). The four laser bays, 128 m long, are situated in pairs on each side of the target chamber. The experiment building (target bay) is a cylinder of 60 m in diameter and 38 m in height. The target chamber consists of an aluminium sphere, 10 m in diameter, fitted with several hundred ports for the injection of the laser beams and introduction of diagnostics. The 240 beams are grouped in 30 bundles of 8 beams in the laser bays and in 60 quads in the target bay. Numerous diagnostic instruments will be placed in the target chamber around the target to record essential measurements. They will make it possible to observe the behaviour of the target during its implosion and at the time of ignition. These diagnostics are the prime tools for the physicists to determine the characteristics of the plasmas they are studying.

LMJ PROJECT STATUS

The building itself is now complete; the mechanical assembly of the bundles in the two first laser bays is achieved and the assembly of the two others is in progress. The LMJ commissioning strategy is to have several steps towards full energy by successively providing each bundle with its optical components and control system, and at the same time using the already commissioned bundles for shots experiments.

A petawatt beam called PETAL is also under construction and will be coupled to LMJ's quad in the next few years.

The first experiments are scheduled on the LMJ facility for the end of 2014.

LMJ CONTROL SYSTEM ARCHITECTURE

General Architecture

The LMJ control system has to manage over 500 000 control points, 150 000 alarms, and several gigabytes of data per shot, with a 2 years on line storage.

It is composed of a dozen of central servers supporting about two hundreds of virtual machines at the central controls level and about 450 PLC's or rack mount PC's at low levels.

All these equipments are distributed on a pyramid of 4 layers: equipment microcontrollers and PLC's on the N0 layer, subsystem supervisory controls on the N1 layer, central controls on the N2 layer and facility management on the N3 layer.

From an industrial point of view, LMJ is divided into a dozen of major contractors, one for each LMJ subsystem. Each of these contractors supplies the N0 and N1 control system levels corresponding to the function he is responsible for. CEA itself defines the common technology and standards and is responsible of doing the integration of the different subsystems.

Software Architecture

All the command control software developed for the N1 and N2 layers by the different contractors uses a common framework based on the industrial PANORAMA E2 SCADA from Codra.

In this framework the facility is represented as a hierarchy of objects called "Resources". Resources represent equipments (motors, instruments, diagnostics…) or high level functions (alignment, laser diagnostics). Resources are linked together through different kinds of relationships (composition, dependency, incompatibility) and the resources life-cycle is described through states-charts. Control-Points, alarms, states and functions can be attached to any resource. Dedicated mechanisms manage the resource reservation and propagate properties and states changes into the tree of resources through relationships. There are about 200 000 resources in order to describe the entire LMJ.

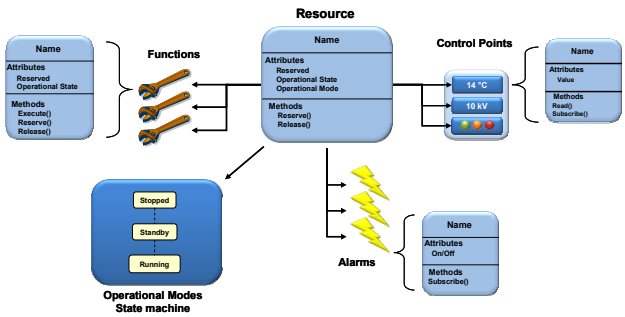


Figure 1: Data model.

The framework implements the data model described above as .net components inside the PANORAMA E2 SCADA and adds some common services to the standard features of PANORAMA E2:

- resources management,
- alarms management,
- lifecycle states management,
- sequencing,
- configuration management,
- event logging

Hardware Architecture

From the hardware point of view the LMJ control system is constituted of two platforms located in two different buildings:

- one for system integration (PFI), which is in operation at the present time in a dedicated building and consists of a clone of the operational control system at the N1, N2 and N3 layers and a mixture of simulators and microcontrollers for representing the N0 layer and real equipments.
- The operational platform (PCI), that will be integrated in the LMJ building at the beginning of next year and which consists of two sub-platforms: a small one for integrating the laser bundles and one for nominal operations.

For each platform, the network architecture (Fig. 2, 3) is composed of two redundant Alcatel-Lucent OmniSwitch 9702E chassis which provide redundant Gigabit attachments to the twelve subsystems backbones and 10 Gigabits attachments to the central controls servers.

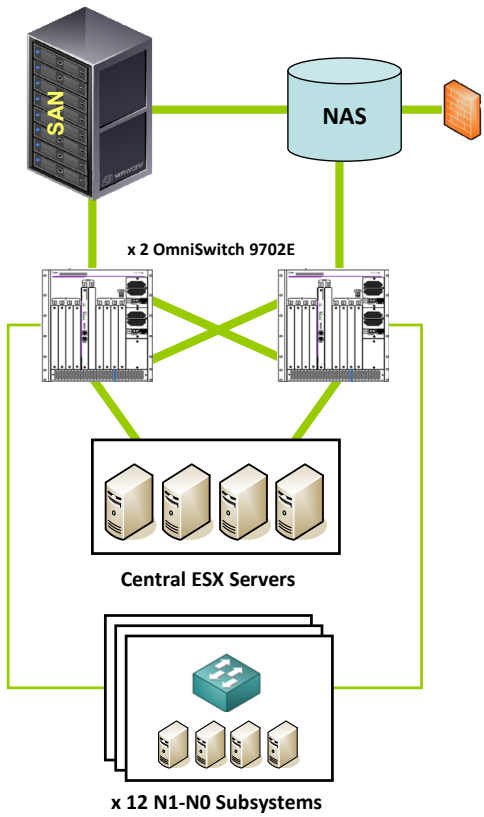


Figure 2: Network architecture.

On each platform, virtual independent contexts are configured using Virtual Routing and Forwarding technologies (VRF): on the PFI this allows to simulate different test contexts at the same time with identical IP address spaces, and on the PCI this allows simultaneous operation from the operational control room and the integration one.

All the N1, N2 and N3 layers are virtualized using VMware and DataCore solutions. Each PFI & PCI platform consists of one virtualization infrastructures composed of:

- 2 DataCore servers, each one managing 12 To of disks,
- 6 to 8 ESX Dell PowerEdge R815 servers, with 4x12 cores and 128 Go of RAM,
- 1 VCenter Server to manage the VMware infrastructure.

Each of these virtualization infrastructures is dimensioned to execute several hundreds of virtual machines.

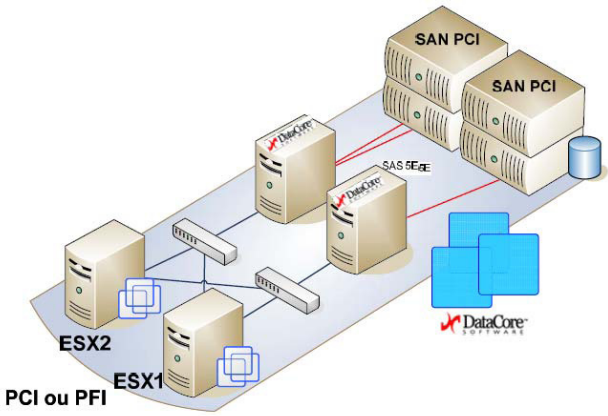


Figure 3: Network architecture.

From the configuration management point of view, application software is generated on the integration platform and stored on a 30 To Network Area Storage (NAS). Images stored on this NAS are made available to the operational platform across a 1 Gigabit dedicated link.

In conclusion the integration and operational platforms of the LMJ control system represent about 500 virtual machines hosted by around fifteen 48 cores / 128 Mo servers.

LMJ CONTROL SYSTEM ROAD MAP

The software framework was commissioned during summer and is now available to contractors to allow them realizing their supervisory systems. The N2-N3 software is currently integrated on a dedicated platform and his commissioning will be accomplished before the end of this year.

The next milestone is to integrate the N2-N3 software with the hardware of the integration platform that will be installed in the PFI building at the end of this year. The integration platform will then be ready to receive control subsystems of the N0-N1 layers from the middle of 2012.

All the N0-N1 subsystems will be integrated to the N2-N3 subsystem on this platform in 2012-2013, before being installed on the operational platform and integrated with laser equipments of the LMJ.

REFERENCES

[1] Michel L. André, "The French Megajoule Laser Project (LMJ)", Fusion Engineering and Design, Volume 44, February 1999 (p.43-49).

[2] J. Nicoloso, J.J. Dupas, J.C. Picon, F. Signol, F. Signol, "LMJ control system Status Report", ICALEPCS 2009, October 2009.

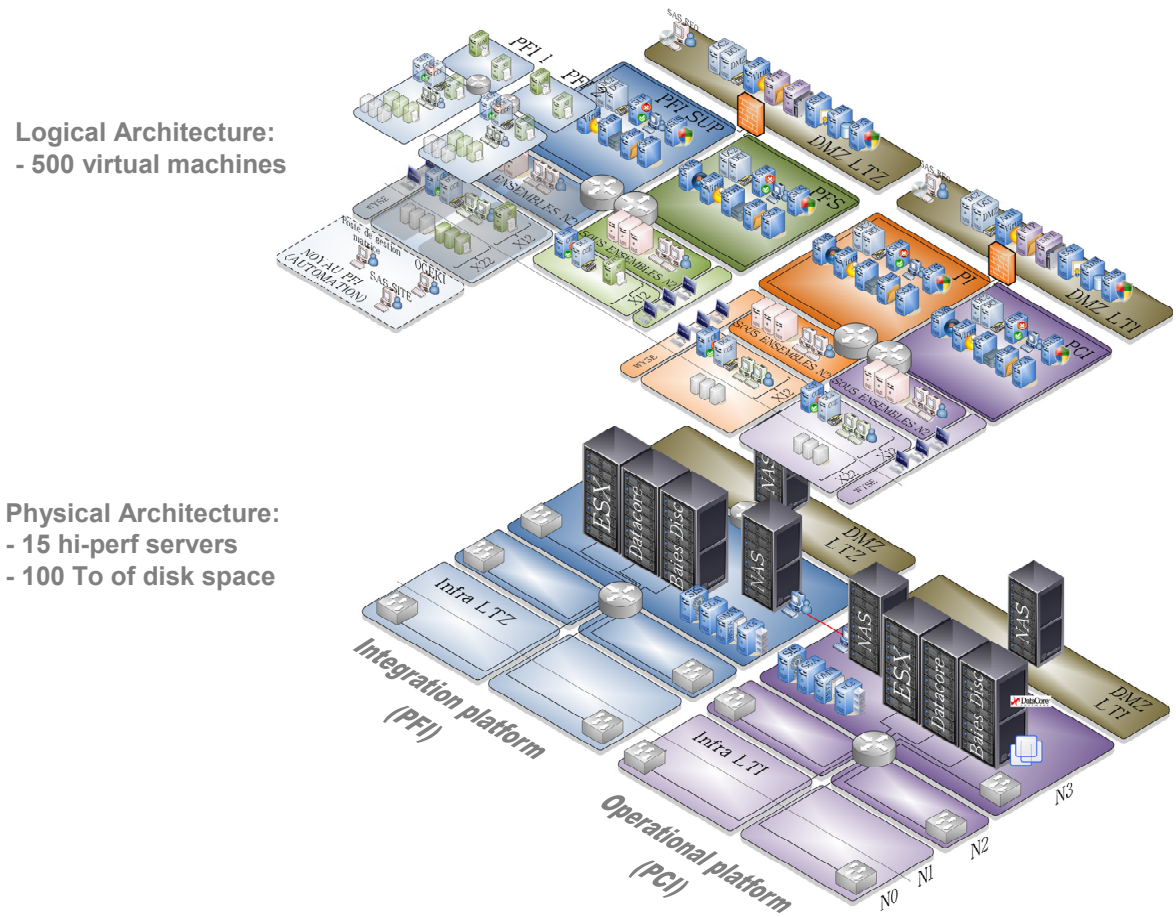


Figure 4: LMJ Central Controls Hardware Platforms, one dedicated for control system integration, one for laser operations.

SUMMARY OF THE 3RD CONTROL SYSTEM CYBER-SECURITY (CS)2/HEP WORKSHOP

S. Lüders*, CERN, Geneva, Switzerland

Abstract

Over the last decade modern accelerator and experiment control systems have increasingly been based on commercial-off-the-shelf products (VME crates, programmable logic controllers (PLCs), supervisory control and data acquisition (SCADA) systems, etc.), on Windows or Linux PCs, and on communication infrastructures using Ethernet and TCP/IP. Despite the benefits coming with this (r)evolution, new vulnerabilities are inherited, too: Worms and viruses spread within seconds via the Ethernet cable, and attackers are becoming interested in control systems. The Stuxnet worm of 2010 against a particular Siemens PLC is a unique example for a sophisticated attack against control systems [1].

Unfortunately, control PCs cannot be patched as fast as office PCs. Even worse, vulnerability scans at CERN using standard IT tools have shown that commercial automation systems lack fundamental security precautions: Some systems crashed during the scan, others could easily be stopped or their process data being altered [2].

The 3rd (CS)2/HEP workshop [3] held the weekend before the ICALEPCS2011 conference was intended to raise awareness; exchange good practices, ideas, and implementations; discuss what works & what not as well as their pros & cons; report on security events, lessons learned & successes; and update on progresses made at HEP laboratories around the world in order to secure control systems. This presentation will give a summary of the solutions planned, deployed and the experience gained.

THE FACT OF ATTACK

2010 has seen wide news coverage of a new kind of computer attack, named "Stuxnet", targeting control systems. Due to its level of sophistication, it is widely acknowledged that this attack marks the very first case of a cyber-war of one country against the industrial infrastructure of another, although there is still much speculation about the details. Worse yet, experts recognize that Stuxnet might just be the beginning and that similar attacks, eventually with much less sophistication, but with much more collateral damage, can be expected in the years to come. Stuxnet targeted a special model of the Siemens 400 PLC series. Similar modules are also deployed throughout the world for accelerator controls like cryogenics or vacuum systems as well as the detector control systems in high energy physics (HEP) experiments.

As with Stuxnet's infection vector, several HEP laboratories reported virus attacks through USB sticks. In one case, the insertion of an infected stick bypassed all firewall protection and network segregation measures put in place to secure the corresponding control system network. As those measures were perceived being sufficient, the affected PC did not run any anti-virus software, which would elsewise have easily quarantined that more than 5-year old virus. However, when trying to establish an IRC connection "home", the virus was quickly identified in the control system firewall logs. Nevertheless, it managed to infect two more control PCs before being fully contained.

In September 2009, one site reported the successful attack against a web server used to display controls information to members of the corresponding experiment's collaboration. For that purpose, all control data was replicated onto a publicly visible web server. However, due to negligence, that web server was neither properly updated nor was the web application properly secured: In a first step, the attacker managed to discover a file injection vulnerability which has subsequently been misused to create a remote shell. A vulnerability in the unpatched kernel subsequently gave the attacker full root access. Its prompt detection avoided further damage.

A similar event was detected at an other U.S. site in spring 2011. After having compromised two Internet facing webservers one month earlier, the attacker escalated privileges right in time for the July 4th holiday week-end. Also here, timely attack detection prevented further misuse.

In April 2011, an email phishing attack hit Oak Ridge National Laboratory (ORNL) in the U.S. Of the approximately 500 recipients of the email about 10% clicked on an embedded link designed to install malware. In one case the user had sufficient privilege that was leveraged to install malware on a large number of additional systems at the laboratory. ORNL chose to sever its connection to the Internet, including blocking web access and external email, to prevent data exfiltration. The clean-up and recovery took two weeks before normal functionality was restored.

The ORNL SNS accelerator controls network was designed to be well isolated from the rest of the ORNL network. Consequently the SNS accelerator was able to remain fully operational during this incident.

The European Organization for Nuclear Research (CERN) was luckier in July 2011 where a researcher on SCADA security found a password of one of CERN's control system. This password was listed in a document made unintentionally public. Before the researcher

* With contributions from E. Bonaccorsi (LHCb), P. Charrue (CERN), P. Chochula (ALICE), S. Hartman (ORNL), T. Hakulinen (CERN), T. McGuckin (JLab), T. Sugimoto (Spring8), F. Tilaro (CERN), V. Vuppala (NSCL/MSU).

published his findings on his personal blog later in August 2011, he informed the U.S. Department of Homeland Security who subsequently informed CERN. Even though this password just provided read access to limited information, it was immediately changed and the compromising document removed from that web-server (as well as from the cache of the Google search engine).

These few security events dismiss the illusion to believe HEP laboratories are not of interest for adversaries and not under attack. Even when all aforementioned security events have been promptly detected, properly analysed, and finally mitigated, acting in retrospect is a bad strategy. While probably not being a high-level target, security officers of HEPs should prepare counter-measures in order to prevent and protect security events from happing.

CHALLENGES IN CONTROL SYSTEM CYBER-SECURITY

However, control system cyber-security is not easy.

CERN's "Access, Safety and Engineering tools" (ASE) group, widely responsible for personnel access and safety systems at CERN, and the ALICE experiment at CERN have both reported about their challenges when implementing standard control system cyber-security measures.

CERN has set up a working group on the protection of control system [4]. Since then, all control systems at CERN must adhere to the "Computer and Network Infrastructure for Controls (CNIC) Security Policy for Controls" [5] following a "Defense-In-Depth" approach. However, over time, some of the access and safety systems have suffered from problems due to the CERN security policies. The most problematic and hardest to debug in the past have been due to the non-robustness of various off-the-shelf systems to periodic security scans. There have also been other problems due to incompatible security patches as well as expiration of service passwords. In the best case, these problems have been an annoyance to the access and safety team. In the worst case, they have prevented the accelerators from starting and personnel from accessing the controlled zones during the very tight maintenance windows.

The ALICE experiment at CERN is experiencing similar problems: Technical requirements and operational constraints [6][7] often directly collide with security measures. ALICE reported that the regular LHC technical stops (2-4 days) are insufficiently short to properly apply all pending security patches. In addition, the CPU consumption of anti-virus software is still one of the top-5 most resource demanding programs.

OVERCOMING THE CHALLENGE

In order to mitigate their problems, CERN ASE has suggested publishing security scan data, i.e., scan schedules, history, and results. This shall allow to correlate scans with the system monitoring data, system failures, and to help prepare for interventions and maintenance. In order to validate the robustness of network-connected devices, CERN's ASE group is considering the TRoIE test-bench, where equipment can be stress-tested and qualified in a controlled environment (see below). The TRoIE test procedure might also be used defining a conformity specification of CERN security measures. The associated requirements could be given to equipment and system vendors during the project definition phase. This document should be sufficiently authoritative and contractual to truly deliver the message to the vendors of the importance of hardening their equipment to the risks of today's computing environments.

Network Segregation, Compartmentalization and Border Control are Essential

The safe and stable operation of the ALICE experiment at CERN is assured by the Detector Control System (DCS), based on a commercial SCADA system PVSS II. The DCS is running on an isolated network, fully compliant with the CERN CNIC standards and rules. The interoperability with external systems is based on the network "exposure and trust" mechanism. DCS hosts can be made visible to external networks by exposing them; remote hosts can be trusted and become accessible from the DCS network. Using these mechanisms, the DCS can largely profit from CERN central computing infrastructure such as name resolution or domain services and reduce the local administrative overhead.

Data produced in ALICE DCS is exchanged with external systems in a secured way. A limited number of data publishers are trusted by the DCS network and central DCS clients can subscribe to the published data using CERN's DIM and DIP data exchange protocols. The DCS then distributes this information to its subsystems using the PVSS II. In the same way, DCS feedback is published to the trusted subscribers. All systems can in addition produce files, which are stored on internal fileservers and automatically mirrored to publicly accessible fileserver located on the CERN general purpose network. Upload of data to the network is subject to strict security policies and is performed by admins upon user's request.

A set of dedicated processes periodically collect condition parameters tagged by detectors for web display and converts them to images which are transferred to a public webserver. The image transfer is based on the concept of private and public fileservers, which assures a complete decoupling of the web services from the DCS infrastructure.

Individual detector control systems are made accessible to remote experts via application gateways, based on Windows Terminal Services. These servers are the only entry point to the system and are accessible only by using user's personal accounts. The use of shared accounts required for the operation is restricted only to the consoles installed in the control room.

The LHCb experiment at CERN pursues as similar road in protecting their control network from malicious remote

access. Operational independence and strong isolation from the Internet as well as from central CERN resources have been important design criteria. Depending on a strong perimeter protection, LHCb has deployed a three-tier redundant firewall providing screened subnets and demilitarized zones. A default deny policy has been implemented together with a set of rules based on the needs of the internal devices to be protected as well as statistical analysis of the boundary and internal network traffic. Each bastion host is hardened at the operating system level, reducing the number of installed applications to minimum. Each web server and reverse proxy has been configured to run as a different domain user and is serving pages from a read only shared network file system whose access is filtered both at OS and network layer. X.509 certificates have been issued by a recognized and "trusted" certification authority and have been installed on all web servers in order to protect the confidentially of sensible data such as usernames and passwords. The entire network traffic is also mirrored and analyzed in real time by an open source intrusion detection system based on Snort.

The Japanese SPring-8 facility serves 55 different beam-lines used by more than 10'000 users per year. The corresponding experimental user networks allow for controlling experimental instruments and data acquisition systems attached to these beam-lines. While SPring-8 has already compartmentalized their experimental user network into 55 segments, it is mandatory to have access to the Internet for data transfer as well as providing users with access to mail and web pages. However, this access is often misused by non-essential applications or inappropriate usage like bandwidth exhaustion by media streaming (YouTube, P2P file sharing), unauthorized instrumental control from outside via VPN, and so on. In particular the latter is prohibited as it collides with SPring-8's radiation safety regulations [8]. Moreover, due to web-browsing, several virus infections had already occurred on the experimental user network.

In order to prevent these threats from spreading to other control systems, SPring-8 had deployed the CheckPoint InterSpect610 intrusion protection system (IPS). While this IPS was suitable in the past, it lacked application signature coverage and was not able to block traffic tunneling via the HTTP web protocol. Hence, in 2010 it has been replaced by the so-called "Next Generation Firewall" from PaloAlto (PA-500 and PA-2050). This firewall can detect and block many (file sharing) applications and viruses including tunnelling protocols. Indeed, until today, this next generation firewall has successfully contained the spreading of 287 different P2P applications as well as 140 different types of viruses. Furthermore, their new firewall provides fundamental statistics for future service upgrades.

The earlier mentioned security event at one particular laboratory forced a full re-examination of their network structure and security. Priorities included isolating and firewalling critical subnets, a thorough segregation of the existing network infrastructure into functional domains, the deployment of application gateways between domains as well as stronger use of administrative and network monitoring tools. Remote access is now based on Virtual Private Networks requiring multifactor authentication (Crypto Cards or USB smart cards). All of these actions are being implemented as part of a more active model of accelerator controls network and system security.

A strong model of Defense-in-Depth is now pursued to produce a long-term solution that meets new requirements while minimizing the impact on-going work at the laboratory and allowing for a continuing cycle of monitoring, assessing and updating security.

DEFENSE-IN-DEPTH

The "Defense-In-Depth" approach requires that security measures have to be deployed on every level of the hardware and software stack, and not only at the network layer. Therefore, a more holistic view is necessary.

Top-Down vs. Bottom-Up

At CERN, the accelerator controls group has started creating a security inventory and risk assessment of the control system computers, devices, accounts and applications, with the goal

- to improve security and reliability of the accelerator control infrastructure;
- to identify the most critical security risks in its control systems; and
- to identify solutions to improve their security, including funding and implementation measures.

This inventory is supposed to summarize all risks using a list of security and reliability attributes like patching status, network configuration, installation base for applications, account usage, etc. While taking advantage of already existing data [9] (and their subsequent clean-up), a web-based "Questionnaire" enables system experts to quickly enter data for the remaining attributes.

The first version has been released to CERN's system experts, and the Questionnaire is currently being populated. In a second step, risk factors will be assigned to each of the attributes in order to determine the overall risk. As a side effect, some experts use this information now for accelerator failure-response and maintenance planning.

Compared to CERN's bottom-up approach, the U.S. National Superconducting Cyclotron Laboratory (NSCL) follows a top-down solution aiming for full compliance with the ISO 27000 standard [10] and final certification [11]. Essential for a successful compliance with ISO 27000 is the full support by management as well as support from all stakeholders involved.

Following this standard, NSCL implemented an Information Security Management System called "ARGUS", a framework of policies, procedures, guidelines and associated resources to achieve the security objectives of the organization. OCTAVE Allegro [12] has been chosen as the risk assessment methodology.

With those tools, the critical information assets, including controls and PLC software & configuration, system documentation, software licenses, etc. have been identified and assigned a relative risk score. This score is combined, categorized and prioritised. The resulting risk is finally mitigated following the controls from ISO 27002 or in-house developed controls. However, acceptance of low or residual risks is possible, too, if properly documented and approved by the management.

Robustness of Controls Devices

Particularly challenging for NSCL have been the hardening of control system platforms such as PLC devices where it was difficult to implement secure software development processes. Focus on such devices has been put on the robustness of industrial control system components by CERN's TRoIE test-bench. Unfortunately, there are no complete and comprehensive security standards yet, which can be followed to secure embedded devices; but several initiatives have been started with the objective of improving the security level and the robustness of industrial systems [13][14].

Therefore, CERN has developed a methodology for automated testing which evaluates the devices' ability to handle erroneous and malicious network traffic. This approach is based on the injection of malformed packets in order to corrupt the normal behaviour of the device and detect possible anomalies. As it is important to enumerate all possible faulty packets for each protocol, TRoIE uses fuzzing and syntax techniques, and lets the tester generating packet sequences in a systematic manner according to the definition of specific protocol syntactic and semantic rules. This technique is generic enough to be applied to any communication protocol, even to industrial ones that exhibit very specific properties and features.

TRoIE is currently under development and a wider publication is under discussion.

CONCLUSIONS

Stuxnet should have been the wake-up call for all those who never believed that control systems could and would be attacked. Indeed, HEP laboratories around the world have seen computer attacks against their facilities, even if these were not dedicated attacks against control systems, yet. However, this should not serve as an argument not to take any action. Continuing to ignore control system cyber-security is grossly negligent.

On the contrary, several HEP labs have started to or do repeatedly review the security protections put in place. Although sometimes cumbersome and difficult to achieve, deploying a "Defense-in-Depth" approach is mandatory and corresponds to good practise. NSCL even goes so far as to aim for full ISO 27000 compliance even on the control system level, a feat which is definitely both an ultimate goal, and a very difficult challenge.

In addition, there was broad consensus that control system cyber-security is more a people problem than a technical one. Establishing a "Security Culture" is needed where system experts, administrators, vendors, and operators cease perceiving "security" as burden but consider it to be an integral part of the system requirements on a par with functional, safety, and maintenance requirements. With such a change of mind-set, a big first step is taken for a better cyber-security of control systems. Subsequent technical steps would then be more easily understood and eventually accepted.

With Stuxnet, a new era has begun. Stay tuned.

REFERENCES

[1] S. Lüders, "Stuxnet and the Impact on Accelerator Control Systems", ICALEPCS, Grenoble, October 2011; these proceedings.

[2] S. Lüders, "Control Systems Under Attack?", ICALEPCS, Geneva, October 2005.

[3] 3rd Control System Cyber-Security CS2/HEP Workshop, http://indico.cern.ch/conferenceDisplay.py?confId=120418.

[4] U. Epting et al., "Computing and Network Infrastructure for Controls", ICALEPCS, Geneva, October 2005.

[5] S. Lüders et al., "CNIC Security Policy for Controls", 2011; https://edms.cern.ch/document/584092.

[6] P. Chochula et al., "Computing Architecture of the ALICE Detector Control System", ICALPECS, Grenoble, October 2011; these proceedings.

[7] A. Augustinus et al., "The Wonderland of Operating the ALICE Experiment", ICALPECS, Grenoble, October 2011; these proceedings.

[8] Y. Furukawa et al., "First Operation of the Wide-area Remote Experiment System", ICALPECS, Grenoble, October 2011; these proceedings.

[9] R. Billen et al., "Accelerator Data Foundation: How it All Fits Together", ICALEPCS, Kobe, October 2009.

[10] ISO/IEC, "Information technology — Security techniques — Information security management systems", ISO/IEC27000:2009, 2009.

[11] V. Vuppala et al., "Securing a Control System: Experiences from ISO 27001 Implementation", ICALPECS, Grenoble, October 2011; these proceedings.

[12] Computer Emergency Response Team (CERT) Program, Software Engineering Institute, "The OCTAVE Allegro Guidebook V1.0", Carnegie Mellon University.

[13] ISA SECURE Certification Program; http://www.isasecure.org/Certification-Program.aspx.

[14] Wurldtech Security Technologies Inc., "The Achilles certification"; http://www.wurldtech.com/cyber-security/achilles-certification.aspx.

SARDANA, THE SOFTWARE FOR BUILDING SCADAS IN SCIENTIFIC ENVIRONMENTS

T. Coutinho, G. Cuní, D. Fernández-Carreiras, J. Klora, C. Pascual-Izarra, Z. Reszela, R. Suñé,
CELLS, Bellaterra, Barcelona, Spain
A. Homs, E. Taurel, V. Rey, E.S.R.F, Grenoble, France

Abstract

Sardana is a software package for Supervision, Control and Data Acquisition in scientific installations. It delivers important cost and time reductions associated with the design, development and support of the control and data acquisition systems. It enhances TANGO [1] with the capabilities for building graphical interfaces without writing code, a powerful python-based macro environment for building sequences and complex macros, and a comprehensive access to the hardware. Just as Tango, Sardana is Open Source and its development model is open to collaboration, which provides a free platform that scales well to small laboratories as well as to large scientific institutions. The first beta version has been commissioned for the control system of Accelerators and Beamlines at the Alba Synchrotron [2]. Furthermore, there is a collaboration in place, comprising Desy [3], MaxIV [4] and Solaris [5], and several other potential users are evaluating it.

THE DESIGN CHOICES IN THE SOFTWARE OF THE CONTROL SYSTEM

Alba is a third generation synchrotron located near Barcelona in Spain. The Installation of the Control System for the Accelerators finished at the end of 2010. The final functional tests took place during the first weeks of 2011. Currently we are carrying out the final stages of the installation and commissioning of the seven Beamlines included in the first phase.

The infrastructure implicated in controls includes more than 350 racks, 6300 equipments and 17000 cables. The controls architecture is highly distributed, comprising about 2500 network devices.

Tango as the Control System Middleware

Tango was chosen among the three options considered: EPICS [6], commercial SCADA (Supervisory, Control And Data Acquisition) and Tango. At that time, the commercial SCADAs were not adapted to the requirements, and although they presented some interesting features off the shelf (like the archiving, trending, etc), many applications needed to be developed anyway, and a non negligible effort needed to be dedicated to integrate motion, synchronization, sequencers, etc. In other words, they were not a solution per se, but to be combined with EPICS, Tango or other toolkits.

Sardana: the Scientific SCADA Suit

The other way around, both Tango and EPICS have various choices as graphical toolkits. However, they both lack in some way an integrated development and runtime environment as commercial SCADAs offer. But on the other hand, commercial SCADAs did not fulfil the requirements either. In several installations, like synchrotrons, we find a large control system for the particle accelerators with different subsystems such as vacuum, radio frequency, power supplies, diagnostics and protection systems, and many "smaller" control systems, one per experimental station. They usually have significantly different requirements.

We needed a flexible graphical interface, allowing multiple clients, with many specific capabilities such as diffractometers control, and above all, a powerful sequencer. Many of these characteristics are implemented in SPEC [7]. However, SPEC has some limitations in the Graphical interfaces and in the capability for managing multi-clients. Therefore at that point we decided to start a development of a SCADA for scientific institutions: *Sardana* [8]. In order to facilitate its adoption, Sardana is built on widely available Open Source technologies and it is itself distributed under the Lesser General Public License [9]. Nowadays it has already been exported outside Alba, and other institutes like Desy or MaxLab are participating in the effort.

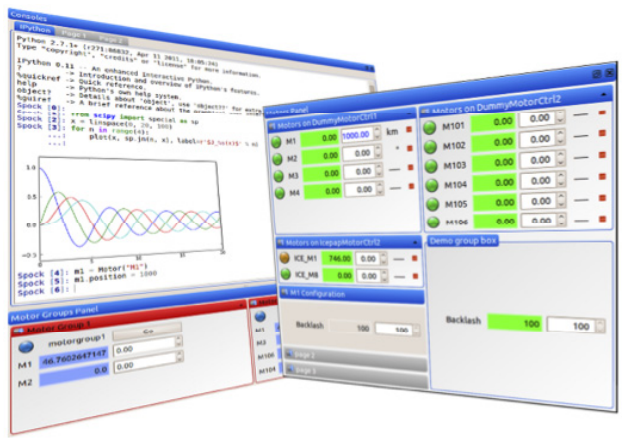


Figure 1: Few views and widgets of the Sardana´s Human-Machine interfaces.

Sardana provides optimized access to hardware, macro execution, software synchronization, generic graphical interfaces, and access to save/restore facilities (Fig. 1). It is mostly developed in Python. It has a Core in charge of managing the hardware: the "*Device Pool*". It has also a powerful sequencer which handles macros, scans, series, loops, etc: the "*Macroserver*" (Fig. 5). This is imperative in any experimental station in a synchrotron, and extremely useful in all cases.

HUMAN MACHINE INTERFACES

The Graphical User Interfaces (GUIs) are developed in a framework called Taurus [10]. It is written in python and built on top of PyTango [11] and PyQt4 [12]. The Command Line Interface (CLI) is based on IPython and makes extensive use of Taurus as well. It is known as "*spock"*(Fig.4), and it was highly inspired on SPEC. The appearance, standard macro names, and the syntax of SPEC command line have been adopted. This is an important point, because many Synchrotron users are familiar with SPEC, considered already a *de facto* standard.

Developers can create Taurus applications from the standard Qt designer benefiting from a catalogue of Taurus widgets which makes the task easier (Figs. 2 and 3).

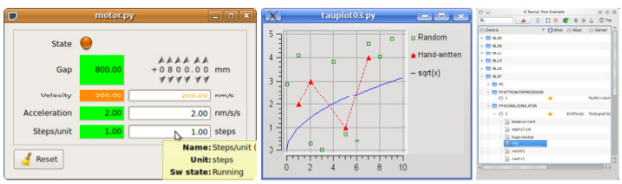


Figure 2: Taurus provides a complete catalogue of widgets.

But the aim is skipping the programming part if it is not needed. Sardana can be setup by "only configuring" the different components. Taurus provides an utility called TaurusGUI which produces standard GUIs from simple configuration files. This utility is typically used to generate standard interfaces for Beamlines. By just configuring a few parameters, a GUI is created for a given Beamline, including widgets for managing and launching macros and sequences, a synoptic view of the system,

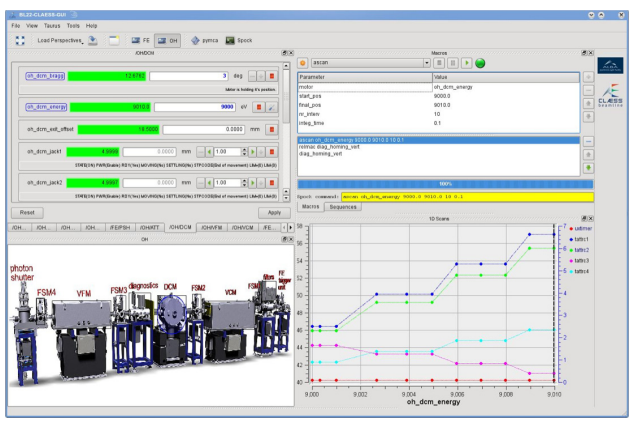


Figure 3: Taurus a highly configurable and customizable standard GUI for Sardana.

panels for controlling the instruments of the Beamline, graphical elements for monitoring values, etc. All these elements are presented in a highly customizable interface which allows the user to create different "perspectives" for different tasks. The elements of the GUI can be re-arranged by drag&drop, and new elements can be added just as easily without leaving the running GUI.

SEQUENCER, MACROS, MOTORS AND SCANS

Sardana provides a standard catalogue of reusable procedures (macros) and sequences for performing specific tasks such as scanning, acquiring data, controlling motors, etc. Besides, it offers the templates and tools for creating and maintaining a user repository of macros.

Although the names of the macros and the syntax might look compatible with SPEC, unfortunately, the differences in the technologies, parsing, grammars and hardware interfaces, made impossible to keep the compatibility with the existing SPEC macros. Hence, SPEC and Sardana are not compatible and macros can not be interchanged between them.

Macros are executed in a central process, called Macroserver. Typically there is a Macroserver in a Sardana installation, although there can be more than one if the application requires it.

Macros are python classes. They are executed and sequenced in the Macroserver, and can also be edited and debugged under its supervision.

Both Taurus based GUIs and CLIs (like spock) connect to the Macroserver through a TANGO object called the "Door". It is through this access point that macro execution can be controlled from outside the Macroserver. A Macroserver can have multiple doors but each door can only run a single macro at a time.

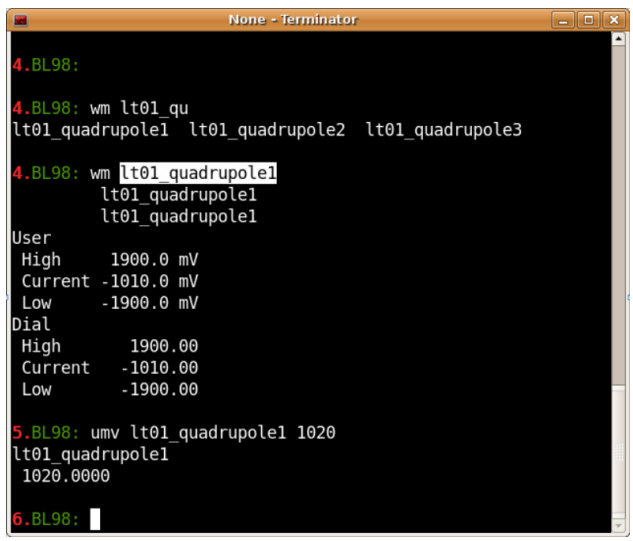


Figure 4: spock: The Command Line Interface of Sardana

When connected to a Macroserver, spock is fed with meta-information about the known macros and elements that are part of Sardana. This way, it can provide features like context sensitive word completion, command history, macro error handling and debugging. Graphical interfaces also receive this information enabling powerful widgets to be constructed .

Macros can be executed from the spock command line, which provides tab completion, history, etc. or from a graphical interface, having, favourites, recently used, etc.

Sequences are batches of macros. They can be easily created from the graphical user interface and also managed as favourites. One does not need to have notions of programming to create a sequence.

THE CORE

The core of Sardana is the so called "Device Pool". It is not only responsible to abstract specific hardware access but also to make sure this access is done as efficiently and as synchronized as possible. Hardware abstraction is possible by providing a set of interfaces to the outside world: Motors (discrete, continuous or pseudo), experimental channels (scalar, 1D and 2D or pseudo) and communication channels. Communication with specific hardware is achieved through the implementation of plug-ins known as "controllers". They can be written in Python or C++. A controller type exist for each interface supported by the Device Pool. When writing a controller, one must obey a specific interface in order for the controller to be considered valid. Controllers can be as simple as a mapping to another tango device server or as complex as a 4C diffractometer.

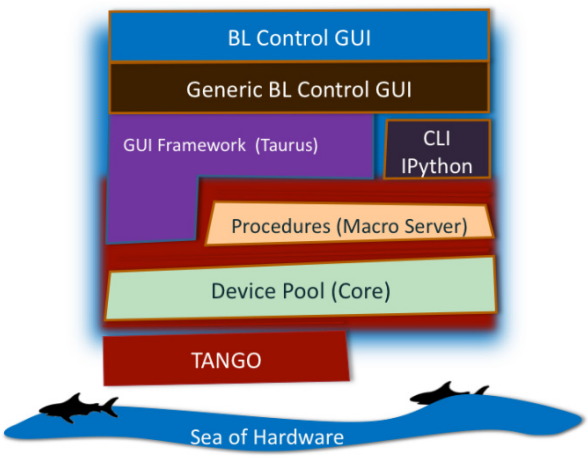


Figure 5: Conceptual design of Sardana.

NEXT STEPS AND STRATEGIC PLAN

In terms of strategy, we could publish the following mission statement: "*Produce a modular, high performance, robust, and generic user environment for control applications in large and small installations",* and the vision: *"Make Sardana the generic user environment distributed in the TANGO project and the standard basis of collaborations in control".*

Although Sardana has been used for the commissioning at Alba, and is installed and running in all Beamlines, is still in an early stage. There is a new release foreseen by the end of 2011, which will improve the experiment configuration, plotting, access to data files and overheads. Looking ahead to the version of 2012, our efforts will be focused among others in the configuration tool, full integration of 2D detectors, and continuous scans frameworks.

CONCLUSION

SCADAs are extensively used in industrial applications. They are optimized for access to Programmable Logic Controllers (PLCs), and popular field buses like Siemens Profibus, etc. Today there is no such a thing for scientific applications for which requirements are drastically different. Labview from National Instruments is a closer example, but it still lacks many important features, like complex "procedures". For example, Beamlines have devices such as diffractometers, combined with detectors, which needed to be scanned with monochromators, which are not well managed by Labview. SPEC manages all these cases, but it has poor Graphical Interfaces, and it does not allow concurrency and arbitration. There is place for improvement in this domain, and Sardana has included all these features in the requirements. There are still a considerable number of features pending and a great effort is still needed in this field. But Sardana is following the correct way, and several other labs (mostly synchrotron) have already expressed their interest.

CONTRIBUTIONS

Many people have worked in this project. In particular J. Ribas, who had a great contribution to the original "pytauico" and "pytauiwi", which was the seed for the current Taurus package. T. Nuñez and T. Kracht at Desy, D. Spruce and K. Larssen at MaxLab for the early tests and contributions to the Sardana device pool and Macroserver.

REFERENCES

[1] http://www.tango-controls.org. The TANGO official website.
[2] http://www.cells.es. The synchrotron light source ALBA. Cerdanyola del Vallès, Barcelona. Spain.
[3] http://www.desy.de. Desy: Deutsches Elektronen-Synchrotron. Hamburg, Germany.
[4] http://www.maxlab.lu.se. MaxLab. National Electron Accelerator Laboratory for Synchrotron Radiation Research, Nuclear Physics and Accelerator Physics. Lund, Sweden.
[5] Piotr Pawel Goryl et al. "Solaris Project Status and Challenges". Solaris Synchrotron, Krakow, Poland.
[6] http://www.aps.anl.gov/epics. EPICS official webpage. Experimental Physics and Industrial Control System.
[7] http://certif.com. SPEC official web page.
[8] http://www.tango-controls.org/static/sardana/latest/doc/html/users/index.html The official Sardana webpage.
[9] http://www.gnu.org/licenses/lgpl.html GNU Lesser General Public License.
[10] http://www.tango-controls.org/static/taurus/latest/doc/html/users/index.html. The official Taurus Documentation
[11] http://www.tango-controls.org/static/PyTango/latest/doc/html/users/index.html. The official PyTango Documentation
[12] http://www.riverbankcomputing.com/software/pyqt/intro PyQt: the official website.

MODEL ORIENTED APPLICATION GENERATION FOR INDUSTRIAL CONTROL SYSTEMS

B. Copy, R. Barillere, E. Blanco, B. Fernandez Adiego, R. Nogueira Fernandes, I. Prieto Barreiro
CERN, Geneva, Switzerland

Abstract

The CERN Unified Industrial Control Systems framework (UNICOS) is a software generation methodology and a collection of development tools that standardizes the design of industrial control applications [1]. A Software Factory, named the UNICOS Application Builder (UAB) [2], was introduced to ease extensibility and maintenance of the framework, introducing a stable meta-model, a set of platform-independent models and platform-specific configurations against which code generation plugins and configuration generation plugins can be written. Such plugins currently target PLC programming environments (Schneider and SIEMENS PLCs) as well as SIEMENS WinCC Open Architecture SCADA (previously known as ETM PVSS) but are being expanded to cover more and more aspects of process control systems. We present what constitutes the UNICOS meta-model and the models in use, how these models can be used to capture knowledge about industrial control systems and how this knowledge can be leveraged to generate both code and configuration for a variety of target usages.

INTRODUCTION

Models are an abstraction of real-world phenomena that represent them for the purpose of problem solving. Model usage is gaining wide acceptance in the domain of industrial control software engineering as it allows to produce implementations that are closer to human understanding of a system than low-level coding.

A meta-model is the "model of a model", describing the properties of the model in a way that makes it applicable to wider range of related problem domains. For instance, a programming language grammar can be considered a meta-model of the programming language it describes. Such a grammar can be used to automatically determine whether programming statements are indeed valid and can therefore result into functional problem resolutions.

Another example of meta-models can be an XML Schema Description (XSD) – which is a meta-model for XML documents, themselves acting as models for the data they contain [3].

THE UNICOS META-MODEL

The UNICOS framework [1], designed to deal with heterogeneous COTS equipment, provides equipment abstractions of one of the following categories :

- I/O Devices – interfaces, which act as data transfer objects.
- Field Devices (pumps, valves etc..) – which act as domain objects.
- Process Control Object (PCO) Devices – which act as controllers, coordinating I/O and field devices.

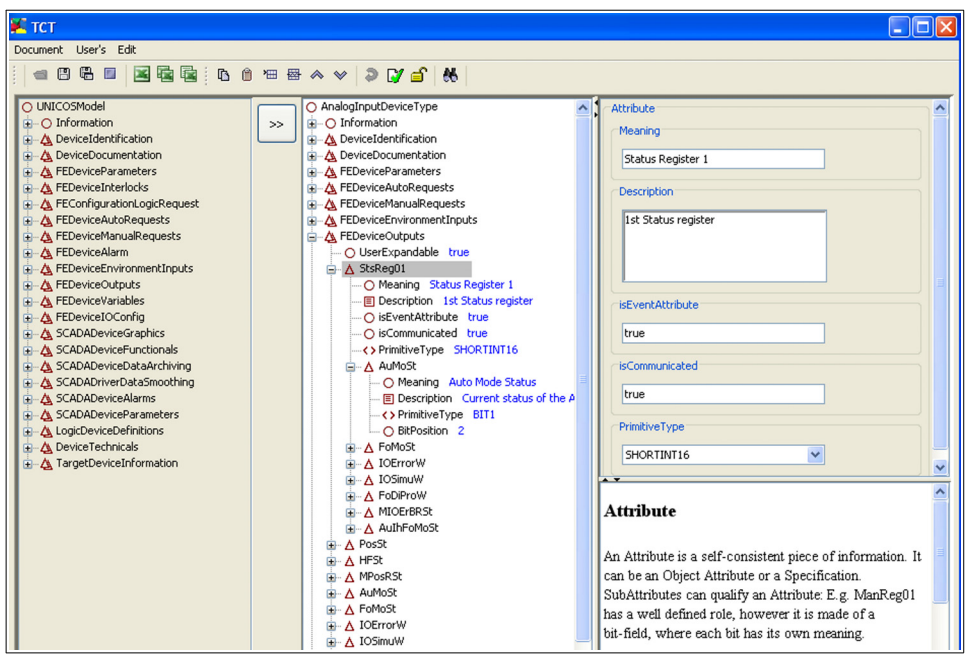


Figure 1 : Screenshot of the FESA General Editor for UNICOS device type creation.

Devices (in effect, object instances) that compose a UNICOS control system all therefore correspond to a Device Type (equivalent to an object class).

The UNICOS meta-model describes the properties and constraints that could be used to specify device types, thereby allowing to provide assistance and formal validation during the specification and implementation of device types.

An XSD Based Meta-Model

The UNICOS meta-model, being expressed in standard XSD, leverages off-the-shelf validation and binding technologies. Being a meta-model for XML documents, it also guarantees that UNICOS device types and device instances (both expressed as XML data) be exploited by any XML aware technology.

In particular, the reliance on XSD to express its meta-model allows UAB to reuse the FESA General Editor [4], a lightweight and general purpose graphical XML editor that provides input completion and validation using an XSD document. The FESA General Editor is used in particular to produce device types that comply with the UNICOS meta-model.

Figure 1 presents a screen capture of the FESA General Editor (also referred to as the "UAB Type Creation Tool" in the UAB context). The UNICOS meta-model is featured on the left hand-side column, where type creators can pick and mix Attributes belonging to different Attribute Families in order to compose their type. The middle column is a user-friendly XML document editor that allows to prepare a UNICOS device type. The right hand-side column provides a meta-data-based property sheet, complete with data validation, inline documentation hints and data entry assistance (such as support for drop-down lists of supported values).

Characteristics of the UNICOS Meta-Model

The UNICOS meta-model supports the definition of Device Types, *i.e.* a generalization of devices such as the ones found at CERN in industrial continuous processes.

Device types exhibit Attributes, which are the definition of device type properties. *For example*, attributes typically have properties such as :

- **Description**: used to document an attribute. , its purpose.
- **PrimitiveType** : specifies the property's data type (FLOAT, TIME, STRING, etc.).
- **isCommunicated**: defines whether the property is remotely accessible (from the SCADA layer for instance).
- **isSpecificationAttribute**: this property is used when the attribute value can be modified by the end-user.

To take an analogy with traditional object-oriented programming, Device Types can be thought of as an Object Class, while Attributes are the class' field members.

For better readability, Attributes of a Device Type are grouped in attribute families, such as *for example* :

- **FEDeviceOutputs**, which identifies a front-end device's outgoing signals.
- **SCADADeviceDataArchiving**, which specifies that SCADA-relevant configuration parameters to log data held in device properties to persistent storage.
- **TargetDeviceInformation**, which holds target platform-specific information (*e.g.* information that varies whether we are targetting a SIEMENS or a Schneider PLC device).

Once a Device Type is defined through the UNICOS meta-model (with the usage of the FESA General Editor [4]), and its peer SCADA and device level implementations are made available, the device type is ready to be handed over to UAB application experts, who will establish lists of devices present in their applications.

If we take back our OO programming analogy, application experts simply provide object instances composing their application, and benefit immediately from all the validation and tooling that the UNICOS meta-model grants them usage of.

MODELLING USAGES IN UAB

First and foremost, the UAB model's most prominent application is the UNICOS-CPC framework [5]. UNICOS-CPC provides a library of device types and associated code generation templates suitable for the implementation of continuous control processes, relying on PLC automata and the SIEMENS WinCC Open Architecture SCADA software package.

Building upon the UNICOS meta-model, a significant number of tools have been added to the UAB tool suite, in order to improve the quality of generated applications or better integrate UAB generated applications with the rest of the CERN infrastructure. These tools take full advantage of the standards compliance of the UNICOS meta-model but provide a stark example of how models can be applied to a variety of contexts and be leveraged with diverse levels of software expertise.

Automated Device Type Documentation

UNICOS Device Types, especially those defined in the UNICOS-CPC device types library typically exhibit a large number of properties that grant the UNICOS framework its support for fine-grained operation (any single property of a device instance can for instance be overridden, insuring that process experts can take over in order to deal correctly with unexpected or abnormal situations without experiencing hindrance).

It is therefore paramount to be able to automatically document device types in a human readable format, in order to assist control system developers in preparing device instances. Since the UAB model relies on XML, it can be transformed by means of XML Stylesheets (XSL), an XML dialect specialized in performing structured

template based transformations. Usage of XSL against UAB XML device type definitions makes generating web-based documentation a natural fit.

Figure 2 below presents an example of a manual Digital Input description diagram, mixed with auto-generated documentation of the device environment inputs

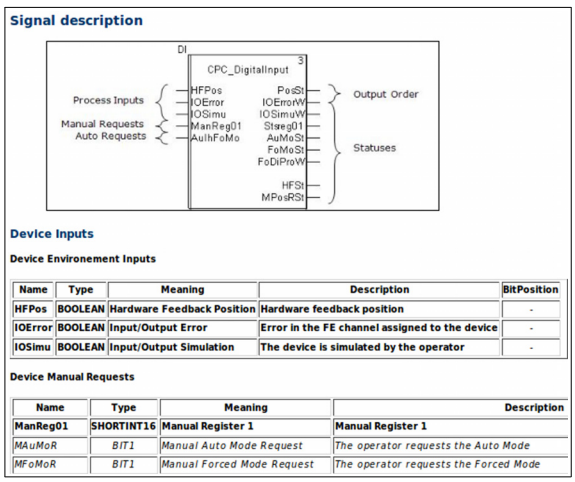


Figure 2 : Screenshot of UNICOS-CPC manual.

Integrating the UNICOS-CPC device type definitions with a content management system such as Atlassian Confluence and its support for inline XSL transformations provides a very simple way to assemble documentation mixing formal technical definitions (such as a list of device pins and their usage) with informal textual descriptions or comments. Other target documentation formats could be DocBook or OASIS DITA, both XML dialects suitable for authoring (and therefore potential targets for XSL transformations), providing results suitable for professional publishing.

Semantic Verification for System Specifications

While the choice of XML schema as a meta-model provides UAB with a large choice of supporting technologies, XSD (intentionally and inherently) lacks the flexibility and richness of other meta-models, such as the OMG Meta Object Facility (MOF) [6]. As a result, certain constraints cannot be easily expressed in XML-friendly form.

Furthermore, UNICOS device types are meant to be preparable by individuals unfamiliar with programming languages or constraint languages such as OCL [6], but still require the possibility to check in details that control system device instance lists comply with formal rules.

To avoid introducing too much complexity in the UNICOS meta-model, the Jython scripting language was chosen to perform these kind of validations, increasing the flexibility of the semantic rules mechanism and allowing the creation of platform specific rules. Most of these rules are applied over the specifications file (this file contains all the devices of a UNICOS application and its parameters).

Examples of existing rules are :

- Check the maximum length of a device's platform dependent name alias.
- Check the relevance of all inter-device relationships referenced in the specifications file.
- Verify that platform dependent input data does not contain illegal characters.

Such rules cannot be easily specified in XML schema.

Extended Configuration Generation

Another example of the application of the UNICOS meta-model lies with the support for extended configurations. UAB generated applications typically need to be integrated with the rest of CERN's infrastructure, such as :

- the LHC Alarm Service (LASER), to inform CCC operators of abnormal conditions across the entire accelerator and technical infrastructure
- other LHC devices communicating through protocols such as CMW (CERN Common Middleware) or DIP (CERN distributed information protocol)

Such integration relies on simple, well-known integration bridges that do not need any advanced code generation, but rather well-known and structure configuration parameters. Using code generation templates for such use cases would therefore introduce unwanted complexity and unnecessarily increase the UAB codebase.

For the purpose of the generation of such configurations, UAB employs a Python-based transformation tool called FlexExtractor [8].

FlexExtractor employs a pluggable strategy to obtain input data (for instance, from XML or a binary Excel file) and produce output data (for instance CSV, or binary Excel file), providing a language independent transformation processor in the middle.

Such a tool is ideal for non-programmers, who benefit from complex transformations without the burden of a programming environment and limitations of XSLT, and allows a decoupling that considerably eases maintenance of configuration generators.

EVOLUTION PERSPECTIVES

While we have seen that an XML Schema-based meta-model and XML data formats open the door to a variety of supporting technologies and target environments, its lack of expressiveness and flexibility also imposes unwanted limitations. Such limitations are the main reason for the creation of tools such as the Semantic Verifier mentioned ealier in this document, delivering the type of validation that XML based ones cannot support at all.

Another important point is that disconnected XML files typically lead to fragmentation of the knowledge and encourage the usage of these XML files in isolated processes.

Since the advent of the UAB project, new technologies have emerged, delivering the same affinities with XML while offering better data management capabilities and superior tooling.

Content Repositories

Content Repositories (also referred to as Java Content Repositories) are an emerging type of object-oriented databases that provide services such as :

- meta-data support – which, as we have seen, is the keystone of any generation or validation process.
- Navigation and query – either based on relational syntax (*i.e.* SQL), XML oriented (*i.e.* Xpath), Object oriented or even free text indexing (*e.g.* fuzzy search engines).
- Serialization support – allowing to import or export tree fragments so that they can be exploited or transformed offline – for instance to an XML representation, or to an in-memory object graph.
- Transaction and versioning support.
- Eventing support – allowing a decoupling between data providers and data consumers.

Some of these services, such as meta-data support, navigation and serialization are already present in the UAB architecture – others such as query, transaction and eventing support are essential in order to let the UNICOS meta-model scale up (for instance, to deal with redundant or hierarchical SCADA architectures, or to handle inter-PLC communication).

Xtext and the Meta Object Facility (MOF)

MOF [6] is a standard for model-driven engineering introduced by the Object Management Group (OMG). It was primarily used to act as a much-needed meta-model for the Unified Modeling Language (UML). MOF can be compared in some respect to XML Schema, but offers much higher flexibility, in its status of a closed meta-model, in effect, a model that can represent itself.

The Eclipse Foundation provides a generation tooling called “Xtext” which accepts models, defined using a closed-meta-model subset of MOF called ECore, but also more traditional inputs such as grammar normal forms or XML schema definitions, and can produce automatically tooling supporting :

- First off, the essential facilities enumerated earlier - meta-data support, navigation, queries, serialization, events.
- Advanced model data entry assistance such as autocompletion, on-the-fly validation and property sheets support, thereby replacing and enhancing the FESA General Editor (written manually) with auto-generated editors.
- Meta-programming APIs that can be used to manipulate data and call methods against the model programmatically at the meta-model level.

CONCLUSION

The UAB technology stack has already demonstrated the importance of a meta-model foundation in its support of the development of code generation tools.

Our experience with tooling development also showed that, even with a meta-model, flexibility and scalability are essential to ensure that the models can evolve and avoid knowledge fragmentation (for instance, rules defined by the Semantic Rule Verifier UAB tool have no place in the UNICOS meta-model today, and the effort of adding them there today would outweigh the benefits).

REFERENCES

[1] R. Barillère, Ph. Gayet, *"UNICOS A Framework to build industry-like control systems, principles and methodology ",* ICALEPCS 2005, Geneva, Switzerland, WE2.2-6I

[2] M. Dutour, "*Software factory techniques applied to Process Control at CERN*", ICALEPCS 2007, Knoxville Tennessee, USA, TPPA03

[3] W3C XML Schema Working Group, “*XML Schema*”, 2000-2007, http://www.w3.org/XML/Schema

[4] M. Arruat et al., "*Front-End Software Architecture*", ICALEPCS 2007, Knoxville, USA

[5] B. Fernandez Adiego, E. Blanco, I. Prieto Barreiro, “*UNICOS CPC6: Automated Code Generation for Process Control Applications*”, ICALEPCS 2011, Grenoble, France, WEPKS033

[6] Object Management Group, “*Meta Object Facility*”, 1997-2011, http://www.omg.org/mof/

[7] Eclipse Foundation, “*Xtext*”, http://www.eclipse.org/Xtext/

[8] R. Nogueira Fernandes, “*FlexExtractor*”, 2010-2011, http://cern.ch/flexextractor

A PLATFORM INDEPENDENT FRAMEWORK FOR STATECHARTS CODE GENERATION

L. Andolfato, G. Chiozzi, ESO, Munich, Germany
N. Migliorini, ENDIF, Ferrara, Italy
C. Morales, UTFSM, Valparaiso, Chile.

Abstract

Control systems for telescopes and their instruments are reactive systems very well suited to be modelled using Statecharts formalism. The World Wide Web Consortium is working on a new standard called SCXML that specifies XML notation to describe Statecharts and provides a well defined operational semantic for run-time interpretation of the SCXML models. This paper presents a generic application framework for reactive non real-time systems based on interpreted Statecharts. The framework consists of a model to text transformation tool and an SCXML interpreter. The tool generates from UML state machine models the SCXML representation of the state machines as well as the application skeletons for the supported software platforms. An abstraction layer propagates the events from the middleware to the SCXML interpreter facilitating the support for different software platforms. This project benefits from the positive experience gained in several years of development of coordination and monitoring applications for the telescope control software domain using Model Driven Development technologies.

INTRODUCTION

Statecharts [1] have been successfully applied at the European Southern Observatory (ESO) for modelling reactive systems [2] such as telescopes and their instruments. Currently all ESO software platforms (the Very Large Telescope (VLT) [10], the Alma Common Software (ACS) [11], and the Standard Platform for Adaptive optics Real Time Applications (SPARTA) [12]), provide different application frameworks based on Statecharts to build monitoring and control applications. For the VLT platform a code generation framework integrated with Rational ROSE and MagicDraw has been developed and successfully employed to create more than 30 C++ control applications for different VLT/VLTI projects [6]. The SPARTA platform offers a C++ library to build Statecharts based applications. The ACS platform provides a tool to transform Statecharts into Java applications using Xpand template language [8].

Table 1: ESO Platforms (for non real-time applications)

Platform	OS	Languages	Middleware
VLT	Linux	C++, TCL/TK	CCS
ACS	Linux	C++, Java, Python	CORBA
SPARTA	Linux	C++	CORBA, DDS

These tools, although implementing the same formalism, support different sets of Statecharts features and, even for the commonly supported features, their operational semantics is often different due to lack of standard semantics for Statecharts [4]. In addition the UML models are not exchangeable between different generators because they are built using different UML profiles. Finally, for all three tools the generated/created source code depends directly on the state machine model and any change in the state machine logic requires a recompilation of the code.

In order to solve the above mentioned problems, reduce maintenance costs, and promote model reusability on different platforms (Table 1), the Generic State Machine Engine (GSME) project started at ESO with the objective of building a platform independent code generation tool from Statecharts called COMODO.

REQUIREMENTS

The requirements for COMODO were derived from the existing tools and can be summarized in:

- Support for the main Statecharts features
- Support for Statecharts inheritance
- Support for graphical and textual representation of Statecharts
- Support for multiple software platforms
- Support for Statecharts interpretation

Looking at the existing telescope control applications based on state machines, the most used Statecharts features are: composite and orthogonal states, shallow and deep history state, guards, entry/exit/on-transition actions, do-activities, and initial/final pseudo-states.

In addition to these standard features a requirement on model inheritance has been introduced to have the possibility of creating models that extend existing Statecharts (for example by adding states and/or transitions). Note that only “extension inheritance” is required, while “refinement inheritance” is not considered [3].

Statecharts is a graphical formalism and since it is part of the UML standard, any UML tool provides the possibility to create, edit, and visualize Statecharts models graphically. However, textual representation is useful for model comparison as well as for scenarios where UML modelling tools are not available. Therefore both graphical and textual Statecharts notations are mandatory for this project.

At last, the ability to change at run-time the behaviour of control applications becomes an important feature for:

- Reducing the model complexity by splitting a large model into several smaller ones. Consider for example the case where a control software application has to support different type of HW components that can be modified at run-time. One possible implementation is to include all HW configurations in one large Statecharts model. A second solution is to have one model per HW configuration and load the correct model at run-time. The latter is usually a better choice in terms of maintainability since smaller Statecharts models are easier to understand.
- Reducing the time needed to apply changes to the state machine logic. For example, during testing/deployment of an instrument, a large amount of time is dedicated to tuning the calibration procedures (i.e. to define in which order to move the hardware and acquire the data). Changing the behaviour of the calibration procedure without the need of recompiling speeds up considerably the testing/deployment phases.

Of course, interpreted Statecharts do not provide the same performances of compiled Statecharts and often require more memory. However, our target applications are monitoring and control applications running on workstations that do not require real-time reaction time.

STATECHART XML

COMODO is based on a StateChart XML (SCXML) engine and a model to text (M2T) transformation tool. Depending on the given software platform configuration, it transforms UML State Machine models into SCXML models and platform specific artefacts such as the application code needed to integrate the SCXML engine and the manually developed code (Figure 1).

SCXML [5] is a standard introduced by the World Wide Web Consortium (W3C) to describe Statecharts. The syntax is based on XML textual notation, and the operational semantics is well defined via pseudo-code. Furthermore, the SCXML language has been designed to be interpreted so that the dynamic behaviour of an SCXML application can be modified at run-time.

SCXML supports all standard Statecharts features required by the project. Moreover, it offers a possible implementation of the Statecharts inheritance via the XML inclusion mechanism.

The support for different ESO platforms is achieved by providing C++ and Java SCXML engine libraries that can be compiled on each of these platforms.

COMODO PROFILE FOR UML

A COMODO application is modelled with a subset of UML which consists of the state machine to specify its behaviour and some elements to describe the interface and its deployment. Applying domain specific stereotypes to the model elements, allows to customize and re-use the model for the different supported software platforms. Stereotypes were collected using the UML profile mechanism in the COMODO profile [13].

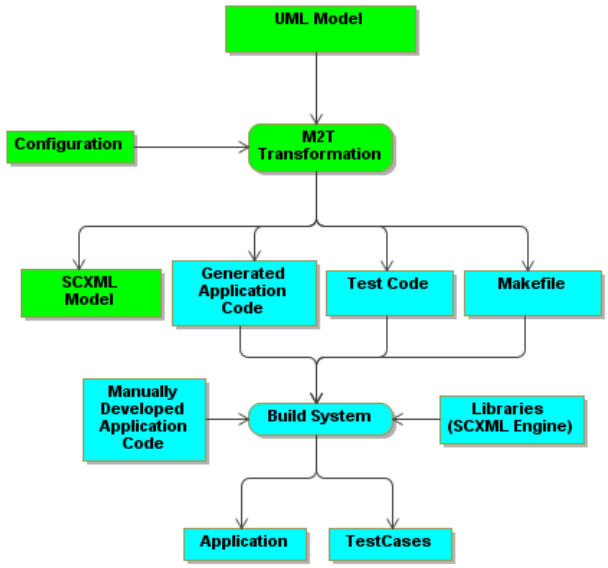


Figure 1: COMODO data flow: in green platform independent activities and artefacts, in blue the platform dependent ones.

MODEL TO TEXT TRANSFORMATIONS

The process to transform a UML model into the target application for a given platform consists of two parts:

- Transformation from UML model to SCXML
- Transformation from UML model to platform specific artefacts

The transformations are based on the mapping between the source model element and the generated artefact.

UML to SCXML Mapping

In order to transform UML Statecharts to SCXML, a mapping between UML model elements and SCXML syntax notation has been defined (Table 2).

The SCXML model is platform independent and even the implementation of entry/exit/on-transition actions and do-activities can be embedded in the model using SCXML scripting action language (or another scripting language supported by the target platform). However our target applications are normally developed in C++ or Java and therefore the interpretation has been limited to the state machine logic while actions and activities are compiled. This approach allows changing the state machine logic at run-time while leaving the advantages of compiled languages for the implementation of actions and activities.

Table 2: UML to SCXML Mapping

UML	SCXML
State Machine	<scxml>
State	<state>
Composite state	<state> (the initial sub-state must be defined)
Region	<parallel> (the initial sub-state must be defined)
Initial pseudo-state	<initial> or initial attribute of <state>
Final pseudo-state	<final>
Entry / Exit action	<onentry> / <onexit>
Transition (trigger [guardExpression] / action)	<transition event="trigger" guard="guardExpression">
Internal transition	<transition> with no target specified
Deep History	<history type="deep">
Shallow History	<history type="shallow">
Activities	<invoke>
Actions	Custom actions
Event (signal)	String
Timer Event	Send command with timeout

UML to Platform Specific Artefacts Mapping

The platform specific artefacts generated from the UML model are:

- Action and Activities stubs (generated only once the very first time)
- The code to handle the platform specific events and propagate them to the SCXML engine
- All artefacts to build the generated application
- Some basic test code used by the automatic test infrastructure provided by the software platforms

Each action and activity defined in the Statecharts model is mapped to a C++/Java class. Activities are started in separate threads while actions are executed via method invocation. It is foreseen to have also an action-to-method mapping (i.e. mapping of a group of actions to methods of a class) to avoid proliferation of classes, facilitate data sharing among actions, and reduce compilation time.

SCXML defines events as simple strings therefore a translation of platform specific events to the SCXML events has to be specified. For example a CORBA call has to be mapped to the corresponding SCXML event string and injected in the SCXML engine. Table 3 shows different types of events supported by the software platforms and their representation in UML.

Table 3: Events Mapping

UML	VLT	ACS	SPARTA
Signal	Command/Reply callback	CORBA method	CORBA method
Time event	VLT Timer callback	ACS Timer	Timer
Signal <<attribute>>	Database change notification callback	CORBA attribute change	CORBA attribute change
Signal <<fileio>>	UNIX file I/O event.	UNIX file I/O event.	UNIX file I/O event.
Signal <<internal>>	Internal event	Internal event	Internal event

IMPLEMENTATION

The UML to SCXML and UML to platform specific artefacts mappings are implemented using EMF [7] and Xpand tool set [8] which include a workflow executor (MWE), the Check constraint language, the Xpand template language, and the Xtend language. The code generator structure is shown in Figure 2.

The MWE workflow is used to drive different steps of the model to text transformations. Four workflows have been created: one per platform and one for the SCXML transformation. The steps of the workflow are:

- Load the UML Profile
- Load the model
- Validate the model by applying Check rules
- Run Xpand templates to generate artefacts

The step validating the model is important since normally UML tools do not impose any rule in the specification of Statecharts while SCXML, being an operational specification, requires well formed models. For this reason a set of Check rules have been written to verify that the model complies with SCXML specifications (for example that the initial state in composite states is always specified or that a transition from the history state is provided).

After the model has been validated, Xpand templates are executed to generate the artefacts defined in the mapping. A set of functions have been developed in Xtend language to help navigating the model to retrieve the model elements properties.

Note that veto strategy or round-trip code generation is not supported by the tool. Customization of the generated code is done via subclassing. Implementation of actions and activities is therefore performed by inheritance from abstract classes generated by the tool.

GENERATED APPLICATION

The applications generated by COMODO use an SCXML engine to process incoming events and trigger transitions.

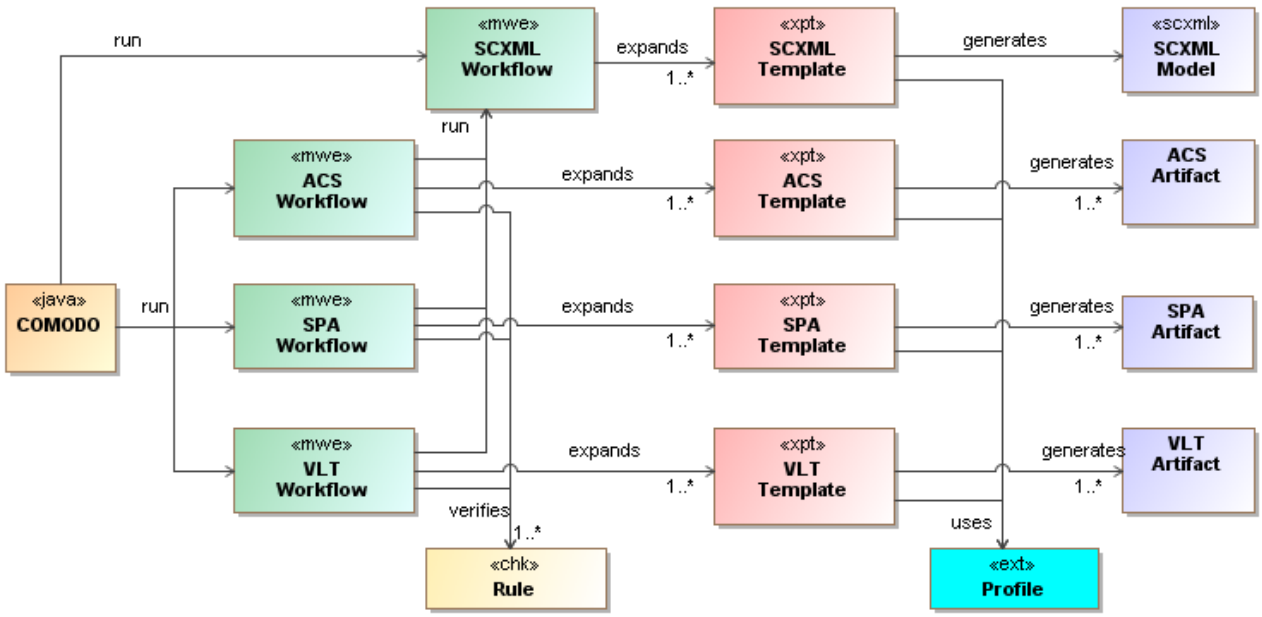


Figure 2: Code generator structure.

Apache Commons SCXML engine [9] has been selected to be the SCXML engine for Java applications. A C++ simplified prototype of the Apache Commons SCXML engine is being developed in house for the C++ applications.

Events are propagated to the SCXML engine via platform specific adapter classes generated by the tool. Actions and activities are executed by the SCXML engine using invoker classes, also provided by the tool. Access to platform services such as logging is provided to actions and activities via injection at creation time.

CONCLUSIONS

In this paper a cross-platform code generator tool from Statecharts was presented. The tool relies on the SCXML notation which provides a standard syntax and semantic specification to describe and execute Statecharts. Since SCXML models are interpreted, generated applications can change behaviour by modifying the state machine logic at run-time without the need for recompilation, reducing therefore the development time.

A UML to SCXML mapping is proposed to benefit from the graphical nature of Statecharts models.

The project is still at prototype level but it provides a complete UML to SCXML transformation and the generation of Java applications for the ACS platform. The next important step is to develop a C++ SCXML engine that is needed to support the other two software platforms.

REFERENCES

[1] D. Harel, "Statecharts: A visual formalism for complex systems", Journal Science of Computer Programming, vol. 8, issue 3, pp. 231-274 (1987).

[2] D. Harel, M. Politi, "Modeling Reactive Systems with Statecharts", McGraw-Hill, (1998).

[3] M. Stumptner, et all, "Behavior Consistent Inheritance with UML Statecharts", Proc. ECOOP'04, p. 59 (2004).

[4] M. L. Crane, J Dingel, "UML vs. classical vs. rhapsody statecharts: not all models are created equal", Software and Systems Modeling, vol. 6, issue 4, pp. 415-435 (2007).

[5] "State Chart XML (SCXML): State Machine Notation for Control Abstraction", W3C Working Draft, April 2011, (2011).

[6] L. Andolfato, R. Karban, "Workstation Software Framework", Proc. SPIE 2008, vol. 7019, (2008).

[7] D. Steinberg et al., "EMF: Eclipse Modeling Framework", 2nd Edition, Addison-Wesley Professional, (2008).

[8] B. Klatt, "Xpand: A Closer Look at the model2text Tranformation Language" 12th European Conference on Software Maintenance and Reengineering, (2008).

[9] Apache Commons SCXML; http://commons.apache.org/scxml/.

[10] K. Wirenstrand, "VLT telescope control software: status, development, and lessons learned", Proc. SPIE 2003, vol. 4837, p. 965 (2003).

[11] G. Chiozzi et all, "ALMA Common Software (ACS), status and development", Proc. ICALEPCS 2009, (2009).

[12] E. Fedrigo et all, "SPARTA: the ESO standard platform for adaptive optics real time applications", Proc. SPIE 2006, vol. 6272, (2006).

[13] G. Chiozzi et all, "A UML profile for code generation of component based distributed systems", this conference, (2011).

LHCb ONLINE INFRASTRUCTURE MONITORING TOOLS

L. Granado Cardoso, C. Gaspar, C. Haen, N. Neufeld, F. Varela, CERN, Geneva, Switzerland
D. Galli, Università di Bologna-INFN, Bologna, Italy

Abstract

The Online System of the LHCb experiment at CERN is composed of a very large number of PCs: around 1500 in a CPU farm for performing the High Level Trigger; around 170 for the control system, running the SCADA system - PVSS; and several others for performing data monitoring, reconstruction, storage, and infrastructure tasks, like databases, etc. Some PCs run Linux, some run Windows but all of them need to be remotely controlled and monitored to make sure they are correctly running and to be able, for example, to reboot them whenever necessary. A set of tools was developed in order to centrally monitor the status of all PCs and PVSS Projects needed to run the experiment: a Farm Monitoring and Control (FMC) tool, which provides the lower level access to the PCs, and a System Overview Tool (developed within the Joint Controls Project – JCOP), which provides a centralized interface to the FMC tool and adds PVSS project monitoring and control. The implementation of these tools has provided a reliable and efficient way to manage the system, both during normal operations as well as during shutdowns, upgrades or maintenance operations. This paper will present the particular implementation of this tool in the LHCb experiment and the benefits of its usage in a large scale heterogeneous system.

INTRODUCTION

Due to the large nature of the LHCb experiment, a high number of PCs is needed to reliably and efficiently operate it. At this scale it is fairly unreliable to monitor and maintain the needed operation parameters without the help of a centralized tool that provides a global overview of the current computer infrastructure status. Furthermore, the heterogeneous nature of the underlying computer infrastructure is an added difficulty for monitoring and control of the experiment computer infrastructure.

The System Overview set of tools, is, therefore, based on a layered approach for monitoring different sets of parameters of the PC infrastructure:

- Hardware Layer – The hardware infrastructure (PCs) where the applications run
- Operating System Layer – The software infrastructure/environment where the applications run
- PVSS Infrastructure Layer – The supervisory control software running the experiment infrastructure/environment
- Application Layer – Specific software needed for system control and operation (PVSS Managers, OPC Servers, etc.)

SYSTEM OVERVIEW SET OF TOOLS

To monitor the parameters for the different monitoring layers, a set of tools were developed, which are able, for each of the existing environments, to gather the relevant parameters:

- Monitoring and Control Servers (FMC Tools)
  - FMC Linux tools [1] – to gather and publish Linux computers data and control processes and IPMI status;
  - FMC Windows Server – to gather and publish Windows computers data;
- GUI (Graphical User Interface) and "aggregation" Tools – to provide a centralized interface for monitoring and control;

Monitoring and Control Servers

The Monitoring and Control Servers gather data from the monitored nodes and publish the monitored parameters via DIM (Distributed Information Management) [2]. These servers are implemented differently for each of the operating system in use in LHCb and can be divided according to the layer they monitor:

Hardware Layer

OS independent:

- ipmiSrv – IPMI Server, which runs in only one LHCb node; This server polls the IPMI status of the PCs in the LHCb network, via ipmitool and publishes this in a DIM Service, which are available for subscription from other tools. The information obtained for each PC is the power status of the PC, and information pertaining the PC sensors which are available via IPMI (temperatures, fan info, currents and voltages)
- vmSrv – which runs on one LHCb node which can act on the hypervisor for the LHCb Virtualization Infrastructure; This server publishes power information for the VMs (Virtual Machines) on the LHCb system and is also able to control the power status via the hypervisor.

OS dependent:
For Linux:

- memSrv – Memory Server, runs on each of the nodes; gathers current memory usage information on the PCs
- cpuinfoSrv – CPU Info Server, runs on each of the nodes; gathers cpu information for the PC (number of cores, speed and type of CPU)

- cpustatSrv – CPU Statistics Server, runs on each of the monitored nodes; gathers CPU usage statistics for the PC
- fsSrv – File System Monitor Server, runs on each of the monitored nodes; gathers info the mounted filesystems and its usage
- nifSrv – Network Interface Monitor Service; monitors network traffic, uptime and statistics for each of the PC Network Interface Cards

For Windows:

- FMC Windows Server - Memory Usage, CPU information, CPU Statistics, File System information, Network Interfaces

Operating System Layer

For Linux:

- osSrv – Operating System Server, runs on each of the nodes; This server gathers operating system and kernel information

For Windows:

- FMC Windows Server - monitors Operating System information, Memory Usage, CPU information, CPU Statistics, File System information, Network Interface information and Processes and Services information.

PVSS Infrastructure Layer

- PVSS pmon process - A process monitor agent linked to each PVSS Project that runs independently from it. This agent monitors and publishes the state of PVSS processes. It can also act on these processes (start/stop/reset)

Application Layer

For Linux:

- tmSrv – Task Manager Server, which runs on each of the nodes; gathers the data for the running processes on each node and is also able to start processes on these nodes.
- psSrv – Process Monitor Server, runs on each of the monitored nodes; gathers info on the running processes on each PC
- pcSrv – Process Controller Server; this server keeps a dynamically manageable list of applications up and running on the farm nodes, stopping and/or restarting them as defined
- logSrv – Log Server, which runs on each of the nodes; collects the logs from the several FMC Linux tools running on the PCs

For Windows:

- FMC Windows Server – monitors processes and services information.

All these servers publish their data via DIM services which are then available for subscriptions from other tools. The services published follow a naming convention, independent of the system where the servers are running.

Note that while the Linux Servers have to run on each of the nodes being monitored, the Windows FMC Server runs on only one machine which has network access to all the windows machines which need to be monitored and the list of machines to be monitored must be defined on start time.

GUI and "Aggregation" Tools

FwFMC

The JCOP Framework [3] FMC component is a PVSS component which subscribes to the DIM services and commands published by the FMC servers and provides a GUI for easy interaction, monitoring as well as an archiving infrastructure for this data.

This tool can be configured from a database which holds the nodes and PVSS projects information as well as from a configuration file. It can also be configured manually adding the nodes and PVSS projects to be monitored.

FwSystemOverview Tool

The FwSystemOverview Tool reutilizes the data subscribed from the FwFMC tool and presents the status of the monitored computers in synoptic panels, with easy control and monitoring possibilities. It is also possible with this tool to monitor single processes and assign alarms for the unexpected dying of these processes. Alarms can also be configured for usage levels of CPU and memory.

The FwSystemOverview tool also has support for monitoring and controlling of PVSS Projects and is able to monitor individual managers via calls to the respective PVSS project Process monitor manager. This allows for PVSS project control for all the running projects from a single place. It is also possible to search for particular managers on all the running projects and act upon them simultaneously.

The monitoring level for each device can be defined and it is possible to monitor only relevant information for each device

FSM (Finite State Machine) [4] objects are also available from this tool, in order to create FSM hierarchies that enable the possibility of grouping the nodes and projects according to desired logical groups. This enables the possibility of grouping and controlling globally nodes and PVSS projects with similar functions or characteristics

LHCb ARCHITECTURE

The LHCb Online System is composed of 1747 PCs, of which 55 machines have Windows and 1692 have Linux installed. Also, of all these machines 34 are virtual machines.

1470 of these machines belong to a CPU Farm to perform the High Level Trigger, 167 are controls machines running the PVSS SCADA system and 110 are for infrastructure support (DBs, storage), webservers, reconstruction or data monitoring

These machines are all connected in the same network and are all accessible to each other within the network.

One machine houses the PVSS part of the System Overview Tool (FwFMC, FwSystemOverview) and provides a DIM DNS Node to where all the running Monitoring and Control Servers publish their data. This machine also serves as IPMI Master.

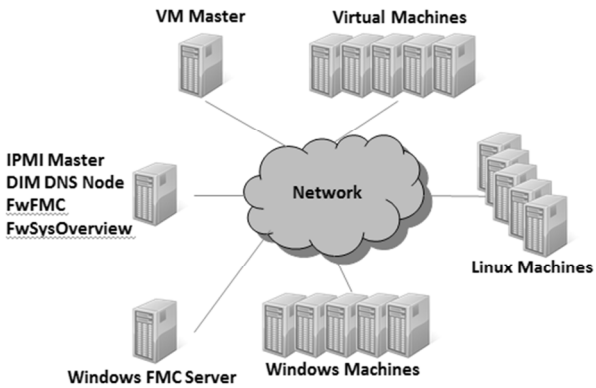


Figure 1: LHCb System Overview Architecture.

For the Hardware Layer monitoring, there are two machines that monitor the power status, fan status, temperatures and voltages:

- 1 IPMI Master – Runs the IPMI Server and polls the IPMI status from each of the nodes present on a configuration file; Controls the power of the configured nodes via IPMI commands
- 1 VM Master – Monitors and controls the power status of the Virtual Machines present on a configuration file.

Currently all of the available machines in the LHCb Online System are configured so as to be possible to monitor and control its power status.

Other parameters from the Hardware Layer (Memory and CPU usage monitoring) are configured on the nodes which are more prone to suffer more of high usage, either by running the respective servers if a Linux Machine or adding the node to a list of monitored nodes on the Windows FMC Server. Alarms are configured on these parameters so that eventual problematic situations are identified as soon as possible.

On the PVSS Infrastructure Layer, all PVSS projects have their status continuously monitored and all the individual managers of the projects are also monitored.

For the Application Layer, all the controls PCs and a few Linux PCs whose processes need to be monitored and controlled more attentively have also the task manager server running. Also the windows PCs which are configured in the Windows FMC server have the processes and services monitoring capability enabled.

LHCb USAGE

LHCb opted to develop the interface based on 2 hierarchies: one for monitoring and control of the hosts and one for monitoring and control of the PVSS projects. These hierarchies are divided according to sub-detector and function. This division allows for, at a quick glance of a synoptic panel, have an overview of the global system as well as the individual sub-detector or any particular function computer infrastructure.

This particular division also allows for the individual sub-detector to access only their specific monitored infrastructure and thus simplifies their particular needs for management.

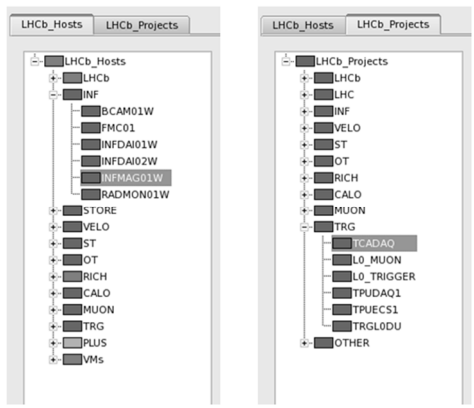


Figure 2: LHCb Hosts and Projects Hierarchies.

Using this division, it is easily identifiable on the "LHCb Hosts" hierarchy if there is any problem with the hardware of a particular system and its diagnose simplified - It is easy to identify the power status and memory usage of the machines as well as a list of running processes on it. It is common to send actions to the hosts (switch ON/OFF, reboot, kill processes) in order to fix the identified abnormalities.

Likewise looking at the "LHCb Projects" hierarchy gives you a global overview of the status of the PVSS SCADA control system and allows you the easy detection of any problem related with the PVSS projects which need to be running to ensure smooth and coherent operation of the experiment. It is commonly used to see if all the required PVSS projects are running and if any particular manager on any of the projects is functioning abnormally (process blocked) or not running (abnormally stopped, killed).

Another advantage of using the System Overview Tool is the possibility to search and globally manage the PVSS managers for all the configured PVSS projects (Fig.3). This allows an easier upgrade of the control software

available on the repositories as all the managers that would need to be stopped and restarted can now be managed from a single access place and in parallel.

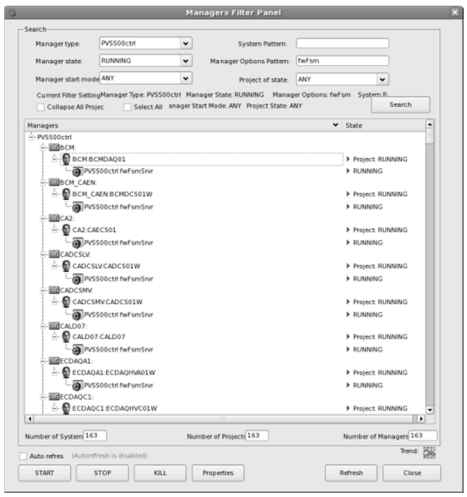


Figure 3: System Overview PVSS manager filtering.

CONCLUSION

The System Overview set of tools provide for a very elegant and user friendly central management tool, with many benefits to the administration of the LHCb online system. It enables an easy and complete monitoring system from a centralized place, permitting a global overview of the system status as well as a fine control over the several nodes, PVSS projects and processes of the system.

The system Overview tool is able to monitor an heterogeneous system with the same interface, detaching the monitoring and control interface from the monitoring and control servers which gather the data and are able to act on the nodes.

The system overview tool is also expandable to TCP enabled devices other than PCs, as soon as there is an available tool for these devices to publish their info into system overview.

REFERENCES

[1] F. Bonifazi et al., "The Monitoring and Control System of the LHCb Event Filter Farm", IEEE Transactions on Nuclear Science, Vol. 55, No.1, Feb. 2008.

[2] C. Gaspar, "DIM - A Distributed Information Management System for the Delphi experiment at CERN", IEEE Real Time Conference, 1993, Vancouver, Canada

[3] O. Holme et al., "The JCOP Framework", ICALEPCS 2005, Geneva, Switzerland.

[4] C. Gaspar and B. Franek, "Tools for the automation of large distributed control systems", IEEE Transactions on Nuclear Science., Vol. 53, NO. 3, June 2006

[5] Prozeßvisualisierungs - und Steuerungs system made by ETM Professional Control GmbH, Eisenstadt, Austria. http://www.pvss.com

[6] F. Varela, "Software management of the LHC Detector Control Systems", ICALEPCS, 2007, Knoxville, Tennessee, USA.

[7] "IPMI v2.0 specifications Document Revision 1.0", http://download.intel.com/design/servers/ipmi/IPMIv2_0rev1_0.pdf

OPTIMIZING INFRASTRUCTURE FOR SOFTWARE TESTING USING VIRTUALIZATION

O. Khalid, A. Shaikh, B. Copy
CERN, Geneva, Switzerland

Abstract

Virtualization technology and cloud computing have brought a paradigm shift in the way we utilize, deploy and manage computer resources. They allow fast deployment of multiple operating system as containers on physical machines which can be either discarded after use or checkpointed for later re-deployment. At European Organization for Nuclear Research (CERN), we have been using virtualization technology to quickly setup virtual machines for our developers with pre-configured software to enable them to quickly test/deploy a new version of a software patch for a given application. This paper reports both on the techniques that have been used to setup a private cloud on a commodity hardware and also presents the optimization techniques we used to remove deployment specific performance bottlenecks.

INTRODUCTION AND BACKGROUND

This paper reports our work to evaluate emerging software technologies such as virtualization and cloud computing for control system applications especially for small teams to quickly setup test environments for development and testing. Virtualization is a software layer that runs on the underlying hardware, and enables system administrators to run multiple operating systems as isolated applications or precisely speaking as virtual machines (VM). The technology have been around since 60's when IBM first developed it to enable users to share mainframe for their applications in an isolated way.

In recent years, virtualization technology have matured to provide bare-metal performance for virtual machines and have been increasingly deployed at large scale (from clusters to data centers) using cloud computing technology to optimize the utilization of the physical infrastructure in an elastic and flexible manner. According to National Institute of Standards and Technology (NIST), Cloud Computing is " a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider's interaction" [1].

NIST identifies two key characteristics of cloud computing that differentiates it from other ways of organizing and accessing computing infrastructures. First is *on-demand self service* that enables a consumer/user to unilaterally provision computing capabilities on demand without requiring human interaction with the service provider. Second feature of cloud computing is *resource pooling* of computing capabilities by the provider to serve multiple user-communities from the same physical infrastructure creating location independence where the user has no knowledge or control over the provided resources. The service model we have opted to deploy is *Infrastructure as a Service* (IaaS) to provide our user community with a capability to provision computing, storage and networking to run any operating system or application.

The key motivation to opt for a private cloud has been the way we use the infrastructure. Our user community includes developers, testers and application deployers who need to provision machines very quickly on-demand to test, patch and validate a given configuration for CERN's control system applications. Virtualized infrastructure along side with cloud management software enabled our users to request new machines on-demand and release them after their testing was complete.

IMPLEMENTATION

The hardware we use for our experimentation is HP Proliant 380 G4 machines with 8GB of memory, 500 GByte of disk and connected with gigabit ethernet. Five servers were running VMWare ESXi bare-metal hypervisor to provide virtualization capabilities [2]. We also evaluated Xen hypervisor [3] with Eucalyptus [4] cloud but given our requirements for Windows VMs, we opted for VMWare ESXi. OpenNebula Professional (Pro) was used as cloud front-end to manage ESXi nodes and to provide users with an access portal [5]. Number of deployment configurations were tested and their performance was benchmarked. The configuration we tested for our experimentation are the following as show in Fig. 1:

- Central storage without front end ($arch_1$): a shared storage and OpenNebula Pro runs on two different servers. All VM's images reside on shared storage all the time.
- Central storage with front end ($arch_2$): a shared storage, using network filesystem (NFS), shares the same server with OpenNebula front end . All VM images reside on shared storage all the time.
- Distributed storage remote copy ($arch_3$): VM images are deployed to each ESXi node at deployment time, and copied using Secure Shell (SSH) protocol by front end's VMWare transfer driver.
- Distributed storage local copy ($arch_4$): VM images are managed by an image manager service which downloads images pre-emptively on all ESXi nodes.

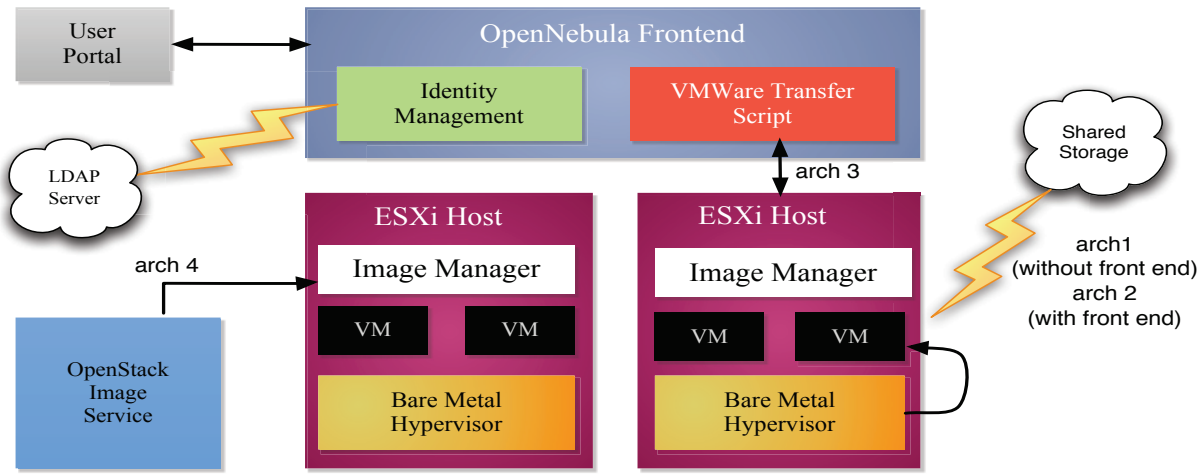


Figure 1: The cloud deployment architecture based on OpenNebula, OpenStack and VMWare ESXi software.

- Front end runs on a separate server and setup VM using locally cached images.

Each of the deployment configuration has its advantages and disadvantages. $arch_1$ and $arch_2$ are using a shared storage model where all VM's are setup on a central storage. When a VM request is sent to the front end, it clones an existing template image and sets it up on the central storage. Then it communicates the memory/networking configuration to the ESXi server, and pointing the location of the VM image. The advantage of these two architectural configuration is that it simplifies the management of template images as all of the virtual machine data is stored on the central server. The disadvantage of this approach is that incase of a disk failure on the central storage; all the VM's will loose data. And secondly, the system performance can be seriously degraded if shared storage is not high performance and doesn't has high-bandwidth connectivity with ESXi nodes. Central storage becomes the performance bottleneck for these approaches.

$arch_3$ and $arch_4$ tries to overcome this shortcoming by using all available diskspace on the ESXi servers. The challenge here is how to clone and maintain VM images at run time and to refresh them when they get updated. $arch_3$ resolves both of these challenges by copying the VM images at request time to the target node (using VMWare transfer script add-on from OpenNebula Pro software), and when the VM is shut then the image is removed from the node. For each new request, a new copy of the template image is sent over the network to the target node. Despite its advantages, network bandwidth and ability of the ESXi nodes to make copies of the template images becomes the bottleneck. $arch_4$ is our optimization strategy where we implement an external image manager service that maintains and synchronize a local copy of each template image on each ESXi node using OpenStack's Image and Registry service called Glance [6]. This approach resolves both storage and network bandwidth issues.

Finally, we empirically tested all architectures to answer the following questions:

- How quickly the system can deploy a given number of virtual machines?
- Which storage architecture (shared or distributed) will deliver optimal performance?
- What will be average wait-time for deploying a virtual machine?

Contextualization

One of the major challenge of deploying windows virtual machines is contextualizing them for a specific environment at deployment time e.g. a windows virtual machine getting a public IP address in CERN network and joining the domain to allow CERN applications to be deployed on it. This requires the VM to part of the public network at the deployment time, and be able to join the domain.

At CERN, network access is controlled by pre-registered list of authorized network interfaces. A list of virtual Machine Access Card (MAC) addresses are pre-registered in the network database. A new VM is deployed with one of the available MAC address. Once its get into a running stage, it gets connected to the network.

Next stage is to configure the machine to acquire the new machine name corresponding to the virtual MAC address. It's simpler to configure for Linux VM's as compared to Microsoft Windows XP VMs. For both we have adopted similar approaches to automate the contextualization process. Each VM is configured to launch a script at boot time that gets its MAC address and compare against a list of registered names (either using a local file or remotely access file). The matching name is updated in the VM and it's rebooted.

For Windows VM, as shown in Fig. 2, before the reboot there is an additional stage of reseting virtual machine's security ID as required by the domain controller and is linked

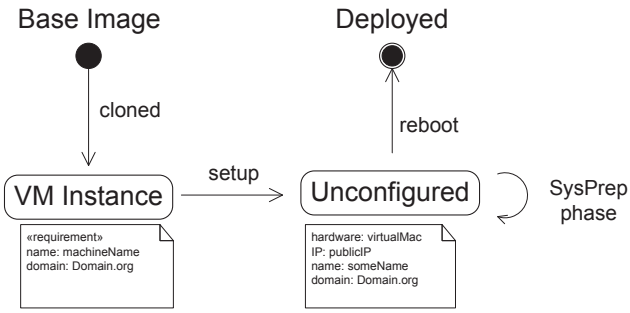


Figure 2: State diagram of a VM starting with the base image, and the stages it goes through to get to running state.

to the active directory entry. If the same security ID is used for all windows VM's, then they can't join the domain. This process is delegated to Microsoft's System Preparation (SysPrep) tool which configures the VM using a local configuration file. Once this configuration is completed; the machine is rebooted and then is available to end-user via the remote desktop connection.

RESULTS

All four different architectures were evaluated for four different deployment scenarios. Each scenario was run three times and the results were averaged and are presented in this section. Any computing infrastructure when used by multiple users goes under different cycles of demand which results in reduced supply of available resources on the infrastructure to deliver optimal service quality.

We were particularly interested in following deployment scenarios where 10 virtual machines were deployed:

- Single Burst (SB): All virtual machines are sent in a burst mode but restricted to one server only. This is the most resource-intensive request.
- Multi Burst (MB): All virtual machines were sent in a burst mode to multiple servers.
- Single Interval (SI): All virtual machines were sent after an interval of 3 mins to one server only.
- Multi Interval (MI): All virtual machines were sent after an interval of 3 mins to multiple servers. This is the least resource-intensive request.

The overall deployment times, as shown in Fig. 3, for $arch_1$ and $arch_2$ are very close to each other for all four test configuration. Both of these architectures were using a NFS based shared storage where all VM images were cloned on a storage server, and only VM setup commands were sent to the ESXi nodes with image pointers. All ten VM's got deployed within 75 mins of request initialization. This is the lower bound of the system. Where as $arch_3$ is using distributed storage but every time a VM request is made, a new image is transferred over SSH which is a very slow process and can take up more then 200 mins for some configurations. $arch_4$ is most interesting that clearly shows the optimization we implemented for auto-deployment of VM images prior to requests in the background which results in VM being deployed within 10mins.

Figure 4 shows that cumulative wait time for VM deployment is fairly stable for $arch_1$ and $arch_2$. $arch_3$ is taking the highest amount of time and $arch_4$ least. Similar pattern of wait time is observable in Fig. 5 and 6 as well. For MI configuration which is least resource intensive, $arch_2$ and $arch_3$ have a similar wait time which shows that both architectures are suitable for small-scale cloud deployments that a VM gets deployed within 30 mins. The only issue is central storage being a single-point of failure incase a hardware fault occurs. $arch_4$ keeps on out perfoming all other configuration, and hence was selected as the target deployment on our private cloud.

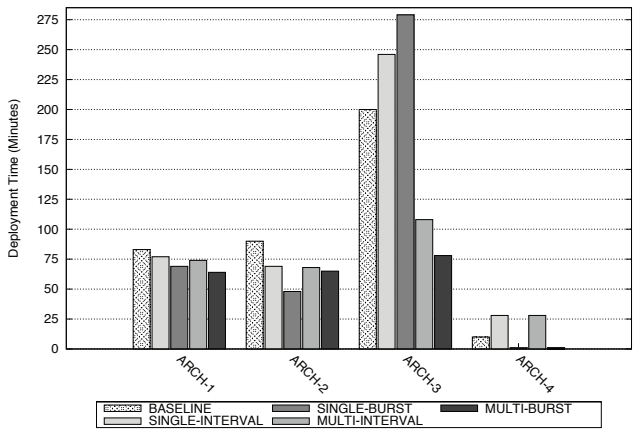


Figure 3: Aggregated total time taken to deploy all VM's in each test configuration for all architectures.

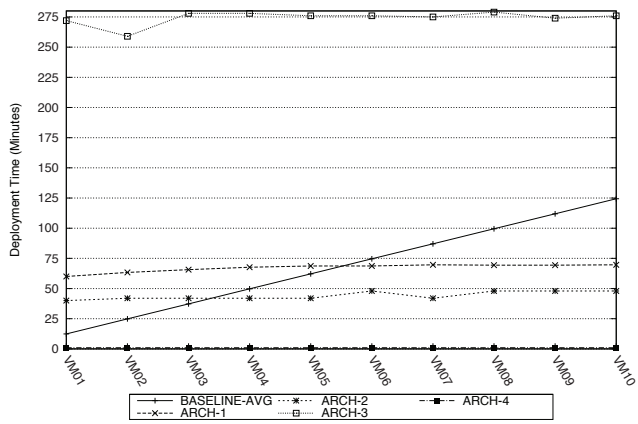


Figure 4: Evolution of deployment time for single-burst (SB) configuration.

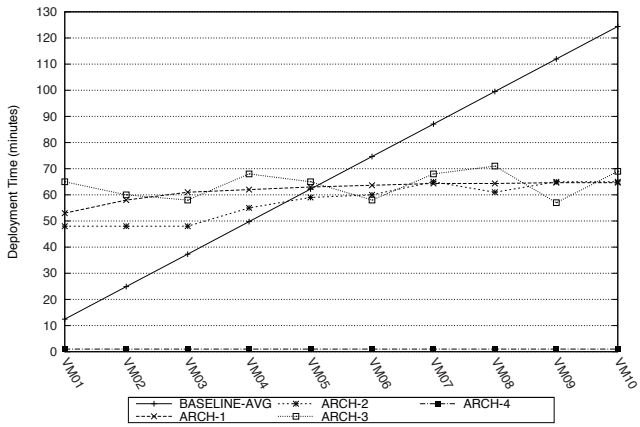


Figure 5: Evolution of deployment time for multi-burst (MB) configuration.

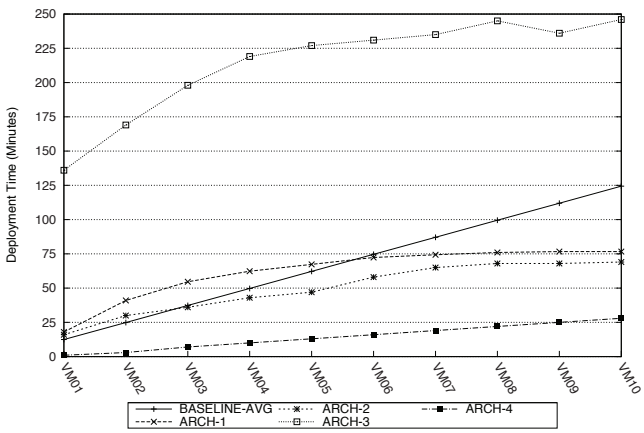


Figure 6: Evolution of deployment time for single-interval (SI) configuration.

CONCLUSION

Virtualization technology and the emerging cloud computing platform provides innovative way to utilize physical infrastructure in an elastic and flexible manner. It enables system administrators to meet various cycles of infrastructure demand when multiple user communities access the same hardware. Without cloud computing, it requires automated system administration tools to provide uniform access to storage, CPU, network and memory of a cluster of machines. Whereas, present day cloud computing management systems allows system administrators to not only virtualize their infrastructure which enables to deploy more applications/machines (virtual) on the shared physical hardware but also reduces the application deployment lifecycle.

In this paper, we have attempted to evaluate small scale private cloud infrastructure (up to 10 hardware servers) for software development teams so that they could quickly and on-demand request machine resources using available virtual machine technology. Our study have highlighted that

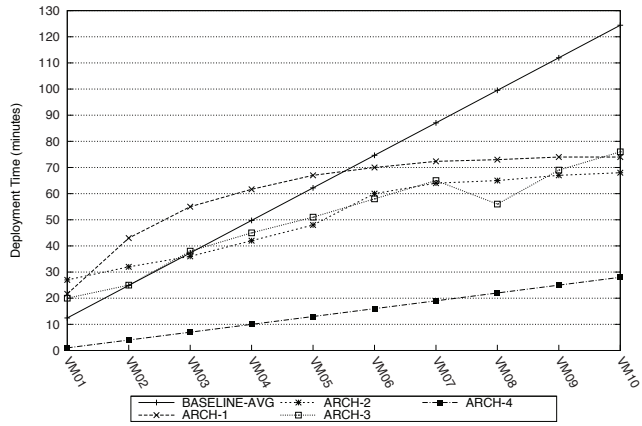


Figure 7: Evolution of deployment time for multi-interval (MI) configuration.

for optimal performance; a high-performance shared storage SAN and high-network bandwidth is preferable but this is often not possible for small scale deployments due to financial constraints. The experiments conducted in this study have indicated that a small scale private cloud is a feasible option without costly SAN or high-speed networking gear.

The results have also shown that distributed storage using locally cached images when managed using a centralized cloud platform (in our study we used OpenNebula Pro) is a practical option to setup local clouds where users can setup their virtual machines on demand within 15mins (from request to machine boot up) while keeping the cost of the underlying infrastructure low.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing", Technical report, National Institute of Standards and Technology, U.S. Department of Commerce, January, 2011.

[2] VMWare, "Migrating to VMware ESXi", *Technical Resource Center*, 2011. `http://www.vmware.com/resources/techresources/10183`

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, Steven, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization", *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP'03, NY, USA, 2003.

[4] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System", *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID'09, Washington, DC, USA, 2009.

[5] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds", *IEEE Internet Computing*, vol. 13, no. 5, pp. 14-22, Sep./Oct. 2009.

[6] OpenStack, "OpenStack: An Overview", *Technical Paper*, 2011. `http://openstack.org/downloads/openstack-overview-datasheet.pdf`

LARGE-BANDWIDTH DATA ACQUISITION NETWORK FOR XFEL FACILITY, SACLA

T. Sugimoto, Y. Joti, T. Ohata, R. Tanaka, M. Yamaga, JASRI/SPring-8, Hyogo, Japan
T. Hatsui, RIKEN/SPring-8, Hyogo, Japan

Abstract

X-ray Free-Electron Laser (XFEL) experiments demands large bandwidth network for data acquisition (DAQ). At the SPring-8 Angstrom Compact Free Electron Laser (SACLA) facility, the experimental data rate is to be up to 5.8 Giga bit per second (Gbps). Some of the experiments demands preprocessing and on-line analysis by high-performance computers. In order to fulfill these requirements, a dedicated network system for DAQ and data analysis has been developed. A DAQ network consists of a dedicated 10 Gbps Ethernet (10GbE) physical layer to secure the data bandwidth and a 1GbE layer for instrument controls. The DAQ network is connected to a primary storage and indirectly to a PC cluster for data preprocessing. A firewall system with virtual private network (VPN) features is also implemented in order to secure remote access from off-site institutes. The preprocessed data plan to be transferred to the off-site supercomputer "K computer" with high efficiency.

OVERVIEW OF SACLA

SACLA is a X-ray free electron laser facility (XFEL) in Japan. The SACLA is located at SPring-8 site, in the west region of Japan. Characteristic specification of the SACLA is electron energy at 8 GeV, peak bunch length of 30 fsec, 18-unit undulator with 18-mm magnetic period. The SACLA is designed to produce X ray with a wavelength as short as 0.06 nm at the repetition rate of 60 Hz. The SACLA consists of accelerator building, undulator building, and experimental facility. To utilize both the SACLA and existence facility "SPring-8", SACLA-SPring-8 experimental facility will be in service in 2012. In the SACLA-SPring-8 experimental facility, X rays from SACLA and SPring-8 can be used simultaneously.

The SACLA facility has successfully established. First X-ray lasing with a wavelength of 0.12 nm was observed on June 7, 2011.[1] By continuing beam commissioning, the wavelength 0.08 nm was achieved on July 13. First test experiment is planned from the October, 2011, and public experimental use will be open on March, 2012.

Experimental Requirement and Data Acquisition Network Design

In March 2012, the SACLA facility will deliver beam time to public users. Multi-Port CCD (MPCCD) detector [2] and its data acquisition (DAQ) system will be used in most of the anticipated experiments. Characteristic requirements are flexibility to change experimental setup within an approximately one-week cycle, robustness to minimize down time, and guaranteed network bandwidth to store all the data. Experimental users also need to use a PC-Cluster system for both on-line and off-line analysis. In order to meet these requirements, we have developed a DAQ system for the SACLA experiments. One MPCCD sensor has 512×1024 pixels with 16-bit data depth. With additional meta data, single-frame data size is about 1 Mega Bytes (MB). At the maximum acquisition rate of 60 Hz, a single sensor produces 480 Mega bit per second (Mbps). Using 12 MPCCD sensors, the data rate reaches 5.8 Gbps. To satisfy the required data rate, we divided the DAQ network (DAQ-LAN) into two physical layers, one is a dedicated 10GbE for large-bandwidth data transfer, and another is a low-latency 1GbE for instrument controls. We applied high-availability techniques to the network trunk lines to reduce down time. The target value of the down time is less than one day per year.

Some of the important SACLA applications, such as three-dimensional (3-D) coherent diffraction imaging at near-atomic resolution and protein nano-crystallography, demand high performance computing (HPC) infrastructures. At the SACLA facility, we plan to connect the DAQ system to the off-site 10-Peta-flops (Pflops) supercomputer "K computer"[3] to make HPC infrastructure available to users. In the foreseen analysis procedure, experimental data are first transferred to an on-site 10-Tera-flops PC cluster with high I/O capability for preprocessing, and then transferred to the K computer via a dedicated network or academic network SINET4.[4] The DAQ-LAN should be connected to the PC cluster and internet, which demands careful network design. This was achieved by the implementation of firewall systems with virtual private network (VPN) features.

THE DAQ NETWORK SYSTEM AT SACLA

The SACLA accelerator facility is controlled by many control-system components. All control-system components are connected to machine control network (CNTL-LAN). Considering how to use the DAQ system and how to ensure the security and the performance, we have developed additional three networks; 1. large-bandwidth data transfer network (DAQ-LAN), 2. experimental user network (DAQ-USER-LAN), and 3. network for data analysis (PC-Cluster-LAN). Policies of these three networks are described in this section.

Since the accelerator is a radiation generator, we must apply strict network security to the CNTL-LAN. Control-system components are fixed during the SACLA operating cycle (typically a few month). On the other hand, experimental machine time is one-week cycle, which is shorter than the operating cycle. The DAQ-LAN should be segregated from the CNTL-LAN for the DAQ-system flexibility. The DAQ system needs accelerator status data, such as wave form, beam intensity, and beam position acquired by non-destructive detectors. To satisfy the requirement and our network security, the CNTL-LAN and the DAQ-LAN are connected each other by a firewall system.

The DAQ-USER-LAN is another network for user experiments. Experimental users can use their own PCs and experimental instruments connected to the DAQ-USER-LAN. The differences between the DAQ-LAN and the DAQ-USER-LAN are 1. all instruments at the DAQ-LAN are managed by the SACLA staff to ensure the DAQ performance (large-bandwidth data transfer and low-latency DAQ control), 2. users' instruments can be connected to the DAQ-USER-LAN by themselves, and the network performance is best-effort 1GbE. The DAQ-USER-LAN is also connected to the CNTL-LAN and the DAQ-LAN by the firewall to acquire accelerator status and to control beamline components.

The PC-Cluster-LAN is another network for the PC-Cluster system. Experimental users can login from the DAQ-USER-LAN, and the users can perform on-line data evaluation using the PC-Cluster system. Off-line users can login from General-LAN in the SPring-8 site. The off-line users can analyze experimental data in the high-speed storage system. To read out the data from storage system, large-bandwidth network is required for the PC-Cluster-LAN. The PC-Cluster-LAN is also used to transfer experimental data from the SACLA to off-site supercomputer "K computer". To meet these requirements, we made policies on the PC-Cluster-LAN; 1. people who have permission can login the PC-Cluster system, 2. login path from the Internet must be protected using VPN system, 3. data-transfer path between the high-speed storage system and the PC-Cluster system should be more than 5.8 Gbps bandwidth, and the bandwidth is upgradable, 4. data-transfer path between the SACLA and the K computer should have more than the DAQ data rate (5.8 Gbps). We have two plans for the data-transfer path to the K computer, one is a best-effort 10 Gbps wide area network (WAN) SINET4, to be connected with in the autumn of 2012. Another plan is data-transfer path using a dedicated line between the SACLA and the K computer to guarantee the bandwidth.

To develop the real network systems, we isolated logical-network layer and physical-network layer. By isolating network layers, the network systems can have upgradability and redundancy. In the next subsections, we show network planning and implementations from each logical and physical view.

Logical Network Planning and Implementations

We made logical network plan of SACLA. We decided IP-address assignment based on same idea of the existence SPring-8 network system.[5] The reasons to adopt the same idea are 1. interconnection method between LANs match each other, 2. to reduce problem such as broadcast traffic[6], 3. to reduce administration cost using same assignment rules. From the IP-address-assignment rule, we applied class-B private addresses to the CNTL-LAN, the DAQ-LAN, and the DAQ-USER-LAN. We also applied class-A private addresses to the PC-Cluster-LAN, because class-A private address indicate open network in the SPring-8.

Each LAN has one (or two with high-availability configuration) layer-three network switch (L3SW). To isolate logical-network layer from physical-network layer, we applied virtual LAN (VLAN) technology. We assigned different VLAN IDs to the large-bandwidth network (10GbE) and the low-latency instrumental-control networks (1GbE). Using VLAN technology, we have flexibility to fold some VLANs on one physical layer or only one VLAN occupies one physical layer such as 10GbE line.

To satisfy requirements on data acquisition and beamline instrumental control on the DAQ-LAN, we assigned twelve VLANs. A segment named "DAQ Data" is a exclusive 10GbE network for data transfer. Other eleven segments are 1GbE networks for experimental instrument control. "DAQ Control" is a 1GbE network to control DAQ components. "Live View" is a best-effort 1GbE network for on-line monitoring. "BL1" (beamline 1), "BL3", and "LH1" (laser hutch 1) are networks to control beamline components such as monochromators and optical mirrors. For future beamline expansion, "BL2", "BL4", "BL5", and "LH2" segments are reserved. Other two VLANs are used for network managements. Near future, network segments for the SACLA-SPring-8 experimental facility will be added to the DAQ-LAN.

Physical Network Implementations for the DAQ System

From the experimental requirements, the physical network should have the following functions; 1. the DAQ-LAN have a bandwidth of 10 Gbps, 2. the DAQ-LAN should be upgraded to 40GbE/100GbE (IEEE802.3ba), 3. the DAQ-LAN can be used several spot in the experimental hall, and 4. the DAQ-LAN and the DAQ-USER-LAN should have robustness to minimize down time less than one day per year. From these requirements, we have developed the physical network in the SACLA experimental facility.

From the requirement of bandwidth and upgradeability, we applied $9/125$ μm single-mode fiber (SMF) as physical trunk lines. For example, the 10GBASE-LR needs two core of SMF, and we can upgrade to 40GBASE-LR4 or 100GBASE-LR4 with existent two-core SMF in the future. If we apply multi-mode fiber (MMF), it is diffi-

cult to upgrade the network, because 40GBASE-SR4 and 100GBASE-SR10 need 8 and 20 core of MMF, respectively.

Figure 1 shows network-wiring diagram at the SACLA experimental facility. The DAQ-LAN consists of two kind of trunk line: one is a exclusive 10GbE for large-bandwidth data transfer (red) and another is versatile 1GbE for instrumental control (blue). We also use same physical trunk line for the DAQ-USER-LAN (purple) and the PC-Cluster-LAN (green). All of wiring lines are distributed from the computer room. Each trunk lines have redundant configuration using link aggregation (LAG, IEEE802.3ad/802.1AX) technology to meet the requirements on robustness of the DAQ-LAN.

The DAQ-LAN with 10GbE is available at several DAQ stations; in the experimental hutch 1–4 (EH1–EH4), 19-inch racks beside EH2 and EH3, and 19-inch rack at experiment preparation room (EP) 1. The DAQ-LAN with 1GbE is available at the LH1, EP5, and the optical hutch 1 (OH1) in addition to the DAQ stations.

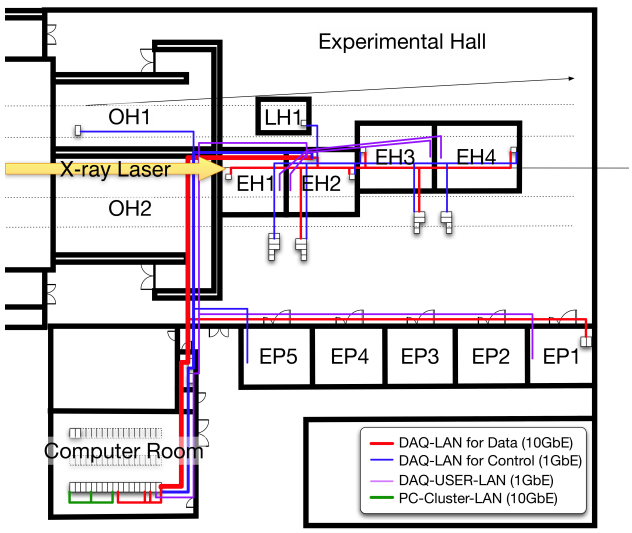


Figure 1: Overview of the SACLA experimental facility. X-ray laser comes from left side of this figure. The X-ray laser is focused in the optical hutches (OHs), and the X-ray laser is used for experiments at the experimental hutches (EHs). Experimental preparation rooms (EPs) are used for instrumental test. Red line is the DAQ-LAN data-transport line (10GbE). Blue line is the DAQ-LAN control line (1GbE). Purple line is the DAQ-USER-LAN (best-effort 1GbE). Green line is the PC-Cluster-LAN (10GbE).

Using the DAQ-LAN, the experimental data flow is described as below. Each detector outputs scattering image at 500 Mbps data rate via the Camera Link interface. MPCCD consists of 12 sensors, total data throughput is to be 5.8 Gbps. VME Camera-Link systems[7] receive data via the Camera Link interfaces. The VME Camera-Link systems send data to data file servers via the DAQ-LAN. Since the trunk line of data-transfer Ethernet is based on 10GbE, the network has enough capacity to transfer experimental data with 5.8 Gbps rate. Digital data from two VME systems are received by one data file server. The data file servers write data on the high-speed storage system via the FibreChannel. Each data file server has three GbE interfaces. Two GbE interfaces are used to receive experimental data at the rate of 500 Mbps from 10GbE line. Another GbE interface is used to control the DAQ system. Thus, we achieved 5.8 Gbps data rate from the MPCCD detectors to the high-speed storage system. Details of the DAQ framework is shown by M. Yamaga et al.[8]

Physical Network Implementation for the PC-Cluster System

We installed a PC-Cluster system at the SACLA experimental facility. The PC-Cluster system is aimed at 1. on-line data preprocessing, and 2. preprocessed data transfer to the K computer. Off-line data analysis will be carried out using off-site 10 Pflops K computer. Generally, network bandwidth over hundreds-kilometer distance is narrow. To make effective use of the K computer, we perform data reduction using the SACLA PC-Cluster system.

The PC-Cluster-LAN requires; 1. large-bandwidth data transfer between the high-speed storage system and the PC-Cluster system, 2. large-bandwidth data transfer from the SACLA to the K computer, 3. robustness to perform on-line analysis, down time should be equal to or less than that of the DAQ-LAN, and 4. versatile use of the PC-Cluster system, and users can login from the DAQ-USER-LAN, the General-LAN, and the Internet. It should be noted that users cannot login the DAQ-LAN from the General-LAN nor the PC-Cluster LAN.

From the requirement of large-bandwidth transfer (1), we have developed the PC-Cluster-LAN between the PC-Cluster system and the high-speed storage as 10GbE network. We also developed the PC-Cluster-LAN between the SINET4 and the PC-Cluster system for the requirement (2). The SINET4 is used for data transfer from the SACLA to users' institutes and the K computer. We also plan to connect the PC-Cluster-LAN with a dedicated network between the SACLA and the K computer to guarantee the data rate to the K computer. From the requirement of robustness (3), we choose ring-topology protocol as a high-availability mechanism for the PC-Cluster-LAN. The ring-topology protocol have much redundancy than that of LAG. From the requirement of versatile use of the PC-Cluster system (4), we installed firewalls between the PC-Cluster-LAN and other LANs to satisfy security policy of the SPring-8 site.

DATA TRANSFER PLAN FROM SACLA

To make effective use of data-transfer network, we have studied large-bandwidth data transfer technique from the SACLA to the off-site supercomputers. We have planed on-line analysis using the K computer. From the result of

on-line analysis at the K computer, we can judge the quality of experimental conditions. We set a present goal to finish the quality-judgement during the beam time. Thus, we can optimize experimental conditions during the experiment.

In order to determine data-transfer bandwidth requirement, we plan to use other two supercomputers. One is "FOCUS" supercomputer located at Kobe, just beside the K computer.[9] The Internet access line of the FOCUS is commercial WAN at 100 Mbps. Another is "e-Science" supercomputer located in the K computer site.[10] The e-Science is connected to the SINET4 at 4 Gbps. The SINET4 between the SACLA and the e-Science can be configured as a 4-Gbps dedicated line. In early of 2012, we plan to evaluate data-transfer throughput of the WAN and a dedicated academic network using the FOCUS and the e-Science.

SUMMARY

We have developed large-bandwidth networks for high-speed DAQ and the PC-Cluster systems at the SACLA. Experimental requirements are 5.8 Gbps bandwidth, flexibility, low down time, accessibility to CNTL-LAN, accessibility to the PC-Cluster system, and network security. We satisfied these requirements, and the DAQ system has no bottle neck from the detectors to the high-speed storage system. In March 2012, the SACLA facility will deliver beam time to public users. MPCCD detector and the DAQ system will be used. In the autumn of 2012, the PC-Cluster system will be connected with the K computer. Using the K computer, we plan to perform XFEL data analysis of 3-D coherent diffraction imaging at near-atomic resolution and protein nano-crystallography.

REFERENCES

[1] H. Tanaka, "Status Report on the Commissioning of the Japanese XFEL at SPring-8", IPAC 2011, San Sebastian, Spain, Septempber 2011.

[2] T. Kameshima et al., "Development status of X-ray 2D Detectors for SPring-8 XFEL", IEEE Nuclear Science Symposium, Knoxville, TN, USA, Octber 2010.

[3] The supercomputer "K computer", Advanced Institute for Computational Science, RIKEN, Kobe, Japan, http://www.aics.riken.jp/.

[4] Science Information NETwork 4 (SINET4), National Institute of Informatics, Japan, http://www.sinet.ad.jp/.

[5] T. Sugimoto et al., "Upgrade of the SPring-8 Control Network for Integration of XFEL", ICALEPCS 2009, Kobe, Japan, October 2009, WED006.

[6] T. Sugimoto et al., "A study of network vulnerability in embedded devices", (CS)2/HEP Workshop, Kobe, Japan, October 2009, Track 08.

[7] A. Kiyomichi et al., "Development of Image Data Acquisition System for Large Scale X-ray 2D Detector Experiments at SACLA (SPring-8 XFEL)", ICALEPCS 2011, WTC Grenoble, France, October 2011, WEPMN028.

[8] M. Yamaga et al., "Event-Synchronized Data Acquisition System of 5 Giga-bps Data Rate for User Experiment at the XFEL Facility, SACLA", ICALEPCS 2011, WTC Grenoble, France, October 2011, TUCAUST06.

[9] FOCUS super Computer, Foundation for Computational Science, Kobe, Japan, http://www.j-focus.or.jp/.

[10] Supercomputer of the e-Science project, Information Technology Center, The University Tokyo, Japan, http://pslogin.cspp.cc.u-tokyo.ac.jp/escience/.

VIRTUALIZED HIGH PERFORMANCE COMPUTING INFRASTRUCTURE OF NOVOSIBIRSK SCIENTIFIC CENTER

A. Adakin, D. Chubarov, V. Nikultsev, ICT SB RAS, Novosibirsk, Russia
S. Belov, V. Kaplin, A. Sukharev, A. Zaytsev#, BINP SB RAS, Novosibirsk, Russia
N. Kuchin, S. Lomakin, ICM&MG SB RAS, Novosibirsk, Russia
V. Kalyuzhny, NSU, Novosibirsk, Russia

Abstract

Novosibirsk Scientific Center (NSC), also known worldwide as Akademgorodok, is one of the largest Russian scientific centers hosting Novosibirsk State University (NSU) and more than 35 research organizations of the Siberian Branch of Russian Academy of Sciences including Budker Institute of Nuclear Physics (BINP), Institute of Computational Technologies, and Institute of Computational Mathematics and Mathematical Geophysics (ICM&MG). Since each institute has specific requirements on the architecture of computing farms involved in its research field, currently we've got several computing facilities hosted by NSC institutes, each optimized for the particular set of tasks, of which the largest are the NSU Supercomputer Center, Siberian Supercomputer Center (ICM&MG), and a Grid Computing Facility of BINP.

A dedicated optical network with the initial bandwidth of 10 Gbps connecting these three facilities was built in order to make it possible to share the computing resources among the research communities, thus increasing the efficiency of operating the existing computing facilities and offering a common platform for building the computing infrastructure for future scientific projects. Unification of the computing infrastructure is achieved by extensive use of virtualization technology based on XEN and KVM platforms. The solution implemented was tested thoroughly within the computing environment of KEDR detector experiment which is being carried out at BINP, and foreseen to be applied to the use cases of other HEP experiments in the future.

INTRODUCTION

Over the last few years the computing infrastructure of Novosibirsk Scientific Center (NSC) located in Novosibirsk Akademgorodok [1] has improved dramatically as the new high performance computing facilities in Novosibirsk State University (NSU) [2] and various institutes of Siberian Branch of the Russian Academy of Sciences (SB RAS) [3] were established in order to be used as shared resources for scientific and educational purposes. The need for providing these facilities with the robust and reliable network infrastructure which would make it possible to share the storage and computing resources across the sites emerged instantly once the facilities have entered production. In 2008 a consortium of the following organizations:

- Institute of Computational Technologies (ICT) [4] hosting all the centralized scientific network infrastructure of SB RAS and Akademgorodok in particular,
- Novosibirsk State University (NSU) hosting NSU Supercomputer Center (NUSC) [5],
- Institute of Computational Mathematics and Mathematical Geophysics (ICM&MG) [6] hosting Siberian Supercomputer Center (SSCC) [7],
- Budker Institute of Nuclear Physics (BINP) [8] hosting a GRID computing facility (BINP/GCF) optimized for massive parallel data processing of HEP experiments (which is supposed to be used as a BINP RDIG [9] and WLCG [10] site in the near future)

was formed with the primary goal to build such a network (named later on as the NSC supercomputer network, or simply NSC/SCN) and provide it with the long term technical support. The first stage of the NSC/SCN infrastructure was deployed by ICT specialists in 2009 and it is being maintained ever since on 24x7 basis.

This contribution is focused on how the existing NSC/SCN infrastructure was used in order to build a virtualized computing environment on top of the NUSC and BINP/GCF facilities which is now exploited for running massive parallel data processing jobs related to KEDR detector experiment [11] being carried out at BINP. The prospected ways of using this environment for serving the needs of other BINP detector experiments and locally maintained GRID sites are also discussed.

NSC SUPERCOMPUTER NETWORK DESIGN AND IMPLEMENTATION

The supercomputer network as it is implemented now, has a star topology and based on 10 Gigabit Ethernet technology. As it is shown in Fig. 1, the central switch of the network is located in ICT and connected to each of the remote participating sites by means of two pairs of SMF G.652 fibers, two of which are equipped with the pair of long range (LR) 10 GbE optical transceivers and the remaining ones are used for two independent 1 Gbps wavelength-division multiplexing (WDM) technology based auxiliary control and monitoring links. The only exception is the link between the ICT and SSCC facilities which is less than 200 meters long and currently deployed over the MMF fiber equipped with the short range (SR) 10 GbE transceivers.

NSC/SCN is by design a well isolated private network which is not supposed to be directly exposed to the general purpose networks of the organizations involved.

#A.S.Zaytsev@inp.nsk.su

Each of the sites connected to the NSC/SCN infrastructure is equipped with the edge switch, used for the centralized access management and control, though the client side connections of the 10 Gbps uplinks are implemented individually on each site, reflecting the architectural differences between them. All the network links are continuously monitored by means of MRTG [12] instances deployed on sites.

The following approaches and protocols are now exploited for exposing computing and storage resources of the interconnected facilities to each other across the NSC supercomputer network:

- OSI Layer 3 (static) routing between the private IPv4 subnets,
- IEEE 802.1Q VLANs spanned across all the NSC/SCN switches,
- Higher level protocols for storage interconnect and also InfiniBand and RDMA interconnect across the sites over the Ethernet links (experimental).

RTT value observed between the sites in the network is less than 0.2 ms. The maximum data transfer rate between the sites for unidirectional TCP bulk transfer over the NSC/SCN network measured with Iperf [13] is equal to 9.4 Gbps. No redundancy implemented yet for the 10 Gbps links and the routing core of the network, but these features are planned to be added during the future upgrades of the infrastructure, along with increasing the maximum bandwidth of each link up to 20 Gbps, while preserving the low value of RTT for all of them. The prospects for extending the NSC/SCN network beyond Akademgorodok are also being considered.

NSC/SCN COMMON VIRTUALIZED COMPUTING ENVIRONMENT

Generic Design Overview

Since the early stages of deployment the NSC supercomputer network is interconnecting three facilities which are quite different from the point of view of their primary field of application and amount of resources available:

- NUSC at NSU: high performance computing (HPC) oriented facility (HP BladeSystem c7000 based solution running SLES 11 x86_64 [14] under control of PBS Pro [15] batch system) provided with 29 TFlops of combined computing power, 108 TB of total storage system capacity, and KVM [16] based virtualization solution,
- SSCC at ICM&MG: HPC oriented facility (30 TFlops of combined computing power plus 90 TB of total storage system capacity),
- GCF at BINP: parallel data processing and storage oriented facility (0.2 TFlops of computing power running SL5 x86_64 [17] plus 32 TB of centralized storage system capacity) provided with both XEN [18] and KVM based virtualization solutions.

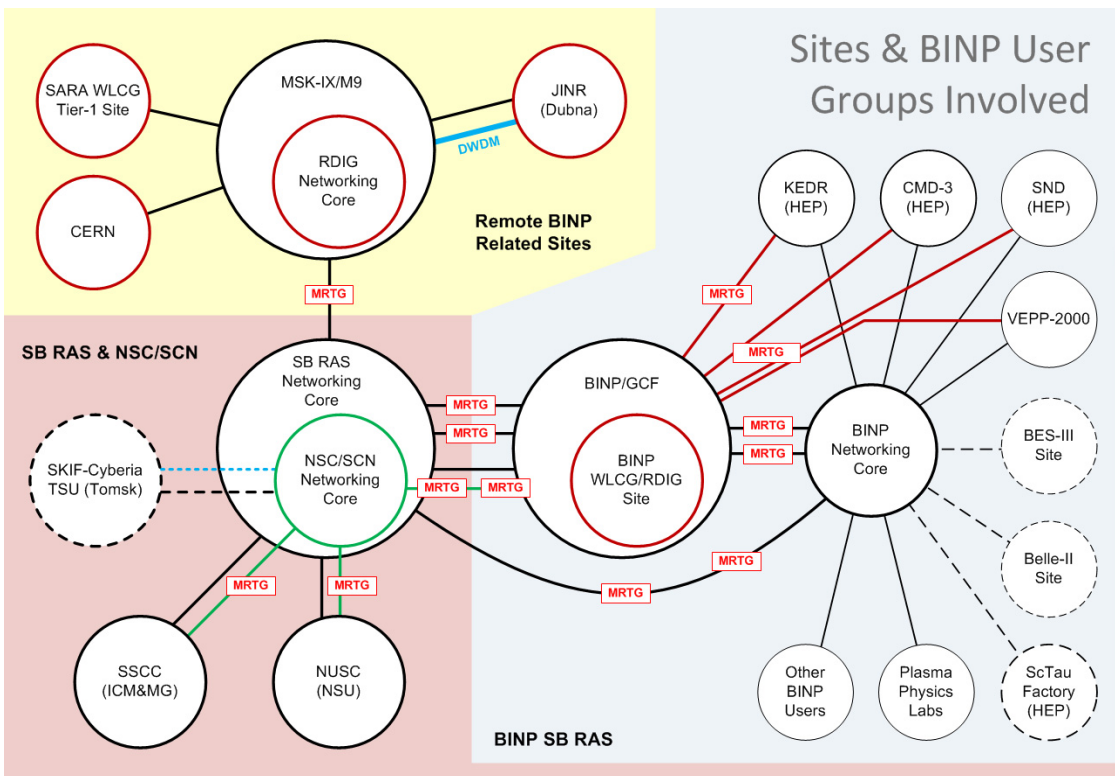


Figure 1: Networking layout of the NSC supercomputer network. Links to the user groups which are still to be established are represented by the dashed lines/circles, and the primary 10 GbE links – by the green lines. MRTG network statistics gathering points are also shown.

Considering the obvious imbalance of computing resources among the listed sites an initiative has emerged to use the NSC/SCN capabilities for sharing the storage resources of BINP/GCF with NUSC and SSCC facilities, and at the same time allow BINP user groups to access the computing resources of these facilities, thus creating a common virtualized computing environment on top of them. The computing environment of the largest user group supported by BINP/GCF (described in the next section) was selected for prototyping, early debugging and implementation of such an environment.

Computing Environment of KEDR Experiment

KEDR [11, 19] is a large scale particle detector experiment being carried out at VEPP-4M electron-positron collider [20] at BINP. The offline software of the experiment was being developing since late 90's. After several migrations the standard computing environment was frozen on Scientific Linux CERN 3 i386 [21] and no further migrations are expected in the future. The amount of software developed for KEDR experiment is about 350 kSLOC as estimated by the SLOCCount [22] tool with the default settings. The code is written mostly in Fortran (44%) and C/C++ (53%). The combined development effort invested into it is estimated to be more than 100 man-years. An overall size of experimental data recorded by KEDR detector since 2000 is 3.6 TB which are stored in a private format. Sun Grid Engine (SGE) [23] is utilized as a standard batch system of the experiment.

All the specific features of the computing environment mentioned here are making it extremely difficult to run KEDR event simulation and reconstruction jobs within the modern HPC environment of the NUSC and SSCC facilities, thus making it an ideal candidate for testing the common virtualized environment infrastructure deployed on top of the BINP/GCF and NUSC resources.

Implementation and Validation Procedures

The following candidates for a solution of the problem stated above were carefully evaluated at NUSC facility while trying to find an optimal configuration of the virtualized environment capable of providing both high efficiency of using the host system CPU power and the long term virtual machine stability at the same time:

- VMware server [24] based solution requiring minimal changes in the native OS of NUSC cluster – *ruled out due to the low performance and poor long term stability*,
- XEN based solution identical to the one deployed on BINP/GCF resources – *ruled out as it required running a modified version of Linux kernel which was not officially supported by the hardware vendor of the NUSC cluster,*
- KVM based solution which have shown the best performance and long term stability while running SLC3 based VMs among all the evaluated candidates and therefore *picked up for the final validation and running the KEDR detector production jobs*.

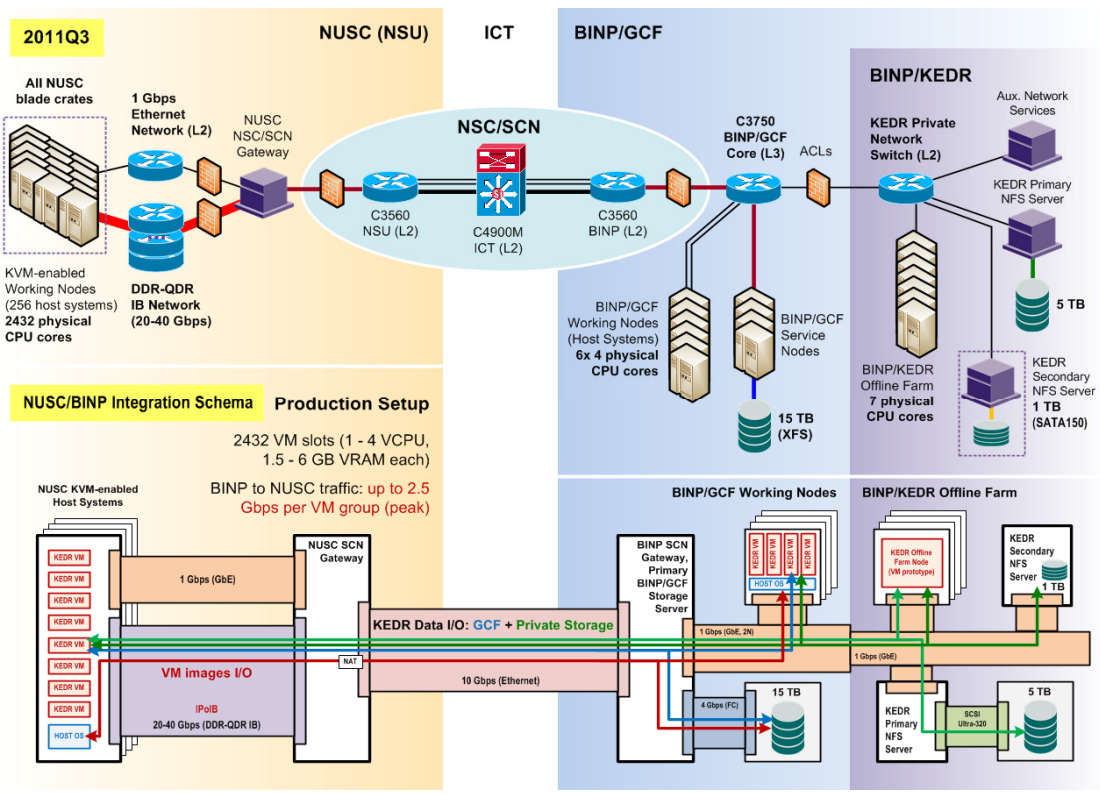


Figure 2: Networking and storage interconnect schema of the common virtualized computing environment spanning across the NUSC and BINP/GCF clusters as it is being used now for running the KEDR detector production jobs.

In addition the KVM based solution was validated during the large scale tests involving up to 512 dual VCPU virtual machines of KEDR experiment running up to 1024 experimental data processing and full detector simulation jobs in parallel controlled by the KEDR experiment private batch system.

All the stages of deployment of the KVM based virtualization environment on the computing nodes of NUSC cluster were automated and now handled via the standard PBS Pro batch system user interface. Furthermore, a generic integration mechanism has been deployed on top of the virtualization system which handles all the message exchange between the batch systems of KEDR experiment and NUSC cluster thus delivering a completely automated solution for the management of the NSC/SCN virtualized infrastructure.

The final layout of networking and storage interconnect schema developed for the KVM based virtualization environment deployed on the NUSC resources is shown in Fig. 2. Note that all the virtual machine images and input/output data samples are exported to the nodes of the NUSC cluster directly from BINP/GCF and KEDR experiment storage systems through the NSC/SCN infrastructure. It is foreseen that the similar solution is proposed to be deployed on SSCC side in 2011Q4 which would result in an increase of the amount of computing resources accessible via the NSC/SCN up to approximately 150 TFlops of computing power and 300 TB of shared storage capacity by the end of 2011 (taking into account the upgrades which are yet to be completed).

CONCLUSION

The supercomputer network of the Novosibirsk Scientific Center based on 10 Gigabit Ethernet technology which was built by the consortium of institutes located in Novosibirsk Akademgorodok, and currently provides a robust and high bandwidth interconnect for the largest local computer centres devoted to scientific and educational purposes. Although nowadays the NSC/SCN infrastructure is geographically localized within a circle of 1.5 km in diameter, it may be extended to the regional level in the future in order to reach the next nearest Siberian supercomputing sites.

The NSC supercomputer network once constructed made it possible to build various computing environments spanned across the resources of multiple participating computing sites, and in particular to implement a virtualization technology based environment for running typical HEP-specific massive parallel data processing jobs serving the needs of detector experiments being carried out at BINP. The solution implemented was tested thoroughly on a large scale within the computing environment of KEDR experiment in 2011Q1 and have been used for running production jobs ever since, delivering up to 75% of computing resources required by KEDR experiment over the last 9 months of continuous operation, resulting in a significant speed-up of the process of physical analysis, e.g. [25, 26].

Recently in 2011Q3 the other two local detector experiments (CMD-3 and SND detector at VEPP-2000 collider [27]) being carried out at BINP have successfully adopted the virtualization solution previously built for KEDR detector in order to satisfy their own needs for HPC resources. The solution obtained is foreseen to be used as a template solution for making a fraction of NUSC computing resources available for deployment of the gLite [28] worker nodes of BINP RDIG/WLCG resource site, Russian National Nanotechnology Network (NNN) resource site of SB RAS maintained by ICT, and also for prototyping the future TDAQ and offline data processing farms for the detector experiment at Super c-Tau Factory electron-positron collider proposed to be constructed at BINP over the upcoming 10 years. An experience obtained while building the virtualization based solution for running production jobs of BINP detector experiments which is compatible with modern high density HPC solutions, such as those deployed at NUSC and SSCC facilities might be of interest for other HEP experiments and WLCG sites across the globe.

ACKNOWLEDGEMENTS

This work is supported by the Ministry of Education and Science of the Russian Federation [29] and grants from the Russian Foundation for Basic Research [30].

REFERENCES

[1] http://en.wikipedia.org/wiki/Akademgorodok
[2] http://www.nsu.ru
[3] http://www.nsc.ru/en/
[4] http://www.ict.nsc.ru
[5] http://www.nusc.ru
[6] http://www.sscc.ru
[7] http://www2.sscc.ru
[8] http://www.inp.nsk.su
[9] http://www.egee-rdig.ru
[10] http://cern.ch/lcg/
[11] http://kedr.inp.nsk.su
[12] http://oss.oetiker.ch/mrtg/
[13] http://sourceforge.net/projects/iperf/
[14] http://www.novell.com/products/server/
[15] http://www.pbsgridworks.com
[16] http://www.linux-kvm.org
[17] http://www.scientificlinux.org
[18] http://www.xen.org
[19] NIM A478 (2002)420-425
[20] http://v4.inp.nsk.su
[21] http://linuxsoft.cern.ch
[22] http://www.dwheeler.com/sloccount/
[23] http://wikipedia.org/wiki/Sun_Grid_Engine/
[24] http://www.vmware.com/products/server/
[25] http://arxiv.org/abs/1109.4205
[26] http://arxiv.org/abs/1109.4215
[27] http://vepp2k.inp.nsk.su
[28] http://glite.cern.ch
[29] http://mon.gov.ru
[30] http://www.rfbr.ru

THE MICROTCA ACQUISITION AND PROCESSING BACK-END FOR FERMI@ELETTRA DIAGNOSTICS*

A. O. Borga#, R. De Monte, L. Pavlovic, M. Predonzani, F. Rossi, G. Gaio, M. Ferianis,
ELETTRA, Trieste, Italy

Abstract

During the construction and commissioning of the FERMI@Elettra Free Electron Laser (FEL) facility, tight requirements for diagnostics readout, processing, and control electronics had been specified; together with a complete integration in the main machine control system. Among the diagnostics devices to be controlled, the Bunch Arrival Monitors (BAM) [1] and the Cavity Beam Position Monitors (C-BPM) [2]. The back-end platform, based on the MicroTCA (or uTCA) standard [3], provides a robust environment for accommodating such electronics, including reliable infrastructure features for slow control and monitoring of the electronics inside the crates. Two types of Advanced Mezzanine Cards (AMC) had been conceived, developed and manufactured in order to meet the demanding requisites. The first is a fast (160 MSps) and high-resolution (16 bits) Analog to Digital and Digital to Analog (A|D|A) Convert Board, hosting two A-D and two D-A converters controlled by an FPGA. The onboard logic is also responsible for service and host interface handling. The latter board is an Analog to Digital Only (A|D|O) Converter, derived from the A|D|A, with an analog front side stage made of four A-D converters. Timing synthesis and distribution from a MicroTCA Central Hub (MCH) slot can be provided by means of a custom MicroTCA Timing Central Hub (MiTiCH), specifically designed for accurate (sub ps range) timing distribution over uTCA backplanes. The overall systems' architectures, together with the AMCs and MCH concept and functionalities, are described hereafter. A summary of the achievements, for each specific use case, together with our experience in the field with the new architecture, are then summarized.

SYSTEMS GENERAL REQUIREMENTS

In spring 2009 a team specialized in digital electronics had been put together by the area leader of the FERMI timing and diagnostics, to develop a common acquisition and processing platform to be interfaced to the machine control system. The goal of the diagnostics uTCA crates, named back-end systems, was to cover the control of 25-30 stations spread along the machine in both the linac and the undulator areas. The ADC count summed up to about 100 inputs, while the DAC output had been estimated to 40 in groups of two fast and two slow converters. Extra general purpose IO capabilities for control and monitor had also been requested.

* Work supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3
# A. Borga, was at Elettra Sincrotrone Trieste S.C.p.A., timing and diagnostics group. He is now at NIKHEF, Electronics Technology department, Science Park 105, 1098XG Amsterdam, Netherlands.

The need for a modular system was therefore obvious. The main facility objective was to lase 19 months later; consequently the development was to be kept realistic, yet innovative because of its tight constraints. After a survey of suitable system architectures, the choice for uTCA was made. Even if very attractive, xTCA for physics was at the time still in its embryonic stage. Designer decided to avoid commercial solutions for two reasons: (i) none of the offers were fulfilling all systems' criteria and the numbers in play wouldn't attract vendors to customize their products; (ii) none had the desired degree of openness required to truly operate, maintain and upgrade such devices. A custom or open design is intrinsically, from hardware to software, under complete control of developers. Costs and manpower were properly scaled to cover the effort required to start with uTCA. The standard had been chosen for its modularity and size, its backplane bandwidth capabilities, and the robustness of modules' supervision. Of fundamental relevance also the fact that MicroTCA had the scalability and the potential for future expansion, characteristics desired by the designers during the conception stage.

BACK-END OVERVIEW

The uTCA back-end system (Figure 1) is a hardware layer interface between the tunnel frontend devices, specific for each diagnostics monitor, and the machine control system. The communication to/from frontends is application dependant; while the data transfer to the control network devices is Ethernet UDP/IP based. Users in the control room can monitor and control devices through Linux real time and Tango servers and panels. Inside the uTCA the clock can be distributed via the backplane, while all other signals are connected via front panel connectors.

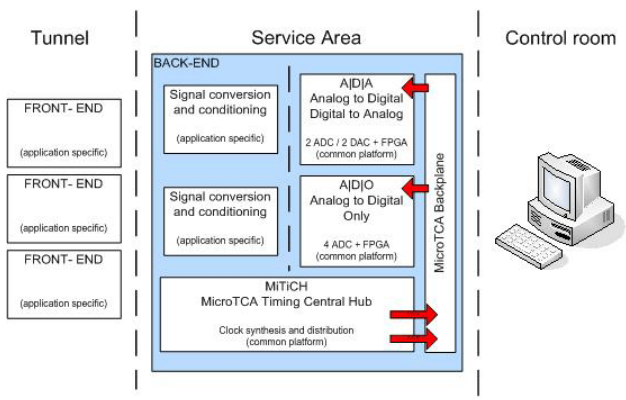


Figure 1: Block diagram of a diagnostic back- end system.

Besides Commercial Off-The-Shelf (COTS) devices like Shelf Managers, CPUs and hard drives; the uTCA back-end crate is conceived to host three types of modules: (i) **A|D|A and A|D|O AMCs**, custom modules described later in this article; (ii) **Side Transition Modules (STM),** interface transition cards (in AMC format) that adapt front-end signals to the inputs and outputs of A|D|A and A|D|O, and communicate with them through fine pitched flat cables; (iii) **Clocking and shelf management MCH**, MiTiCH like modules for card monitor and/or clock distribution.

A|D|A MODULE ARCHITECTURE

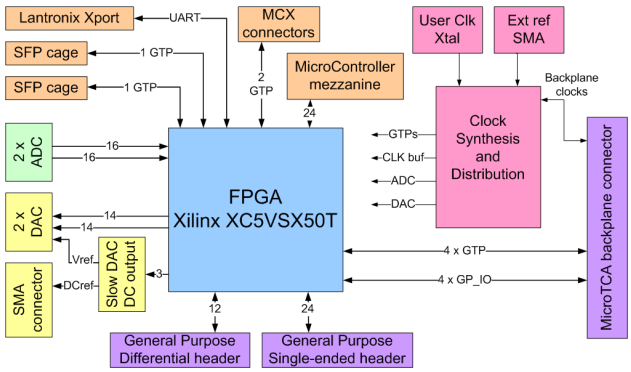


Figure 2: A|D|A AMC module block diagram.

The main purpose of the Analog to Digital and Digital to Analog (A|D|A, Figure 2) converter AMC [5] is acquisition, processing, and generation of fast analog signals. The architecture is based on a Xilinx Virtex-5 FPGA responsible for the data handling of all the external inputs and outputs, the data processing, and the interface to the control system. Extensive board documentation can be found in [6].

A|D|O MODULE ARCHITECTURE

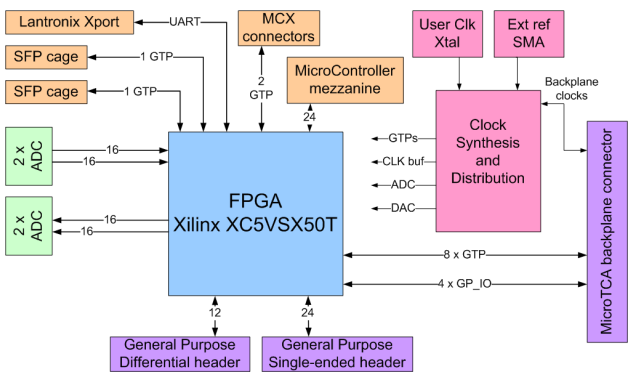


Figure 3: A|D|O AMC module block diagram.

The Analog to Digital Only (A|D|O, Figure 3) converter AMC [7] is derived directly from A|D|A. It is based on the same FPGA responsible for the same generic tasks. The main difference between the two cards is the front-end that in the A|D|O case in made of 4 A-D converters. The backplane connectivity and the clocking schema has both been enhanced and improved.

MITICH MODULE ARCHITECTURE

MCH Carrier Board

The uTCA standard specifies [3] the delivery of power on a backplane connector (tongue 1) physically separated from the one devoted to clock distribution (tongue 2). The development of a base board to provide power to the clock card was therefore necessary (Figure 4). The availability of space on the base board allowed the implementation of additional test features, extender like, for backplane high-speed connectivity characterization. Evaluation of the I^2C control bus for Intelligent Platform Management Interface (IPMI) [4] used for shelf management in uTCA systems had been also predisposed.

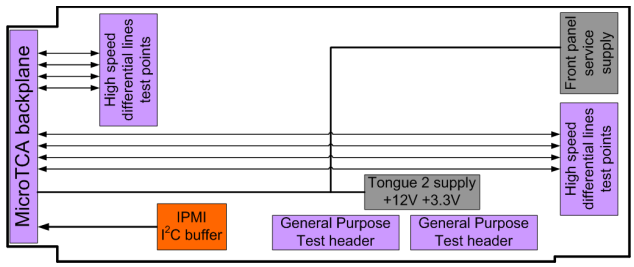


Figure 4: MiTiCH MCH tongue 1 block diagram.

MiTiCH Block Diagram

MiTiCH is a MicroTCA Timing Central Hub (Figure 5) [8] compliant to the uTCA mechanical specifications for MCH tongue 2. Its purpose is the synthesis and distribution of clock signals. To enhance its flexibility, a future revision of the card will host a vector modulator for RF phase control up to S-Band. The key component is an AD9516-4 frequency synthesizer, controlled by a microcontroller, interfaced to the network via a Lantronix. Two stages of fanout buffers distribute two frequencies to up 12 AMC modules on the backplane.

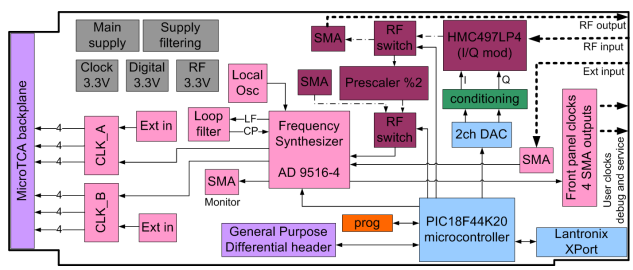


Figure 5: MiTiCH MCH tongue 2 block diagram.

The Charge Pump feedback filter circuit of the PLL can be tuned onboard changing its passive elements; alternatively the internal VCO can be controlled with an external voltage. Characteristics of the circuit can by comfortably studied with the Analog Devices ADIsim-CLK™ tool. Frequencies can be derived either from internal oscillators or external references.

FIRMWARE AND CONTROL SOFTWARE

The idea behind the **firmware** architecture (Figure 6) for both the A|D|A and the A|D|O cards is to provide black box core logic to support a user space containing application specific functions. Three blocks had be identified and developed separately. (i) The control and monitoring block interfaces the card to the controls network. Data flows through an internal memory mapped RAM based space, providing an independent and well defined separation between the different blocks. (ii) The interface layer consists of a series of buffers controlled by state machines collecting and storing data to and from the digitizers. (iii) The clock and timing section exploits digital clock management.

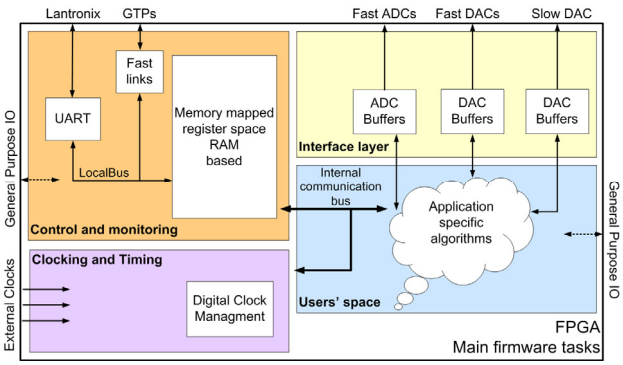


Figure 6: block diagram of the firmware sub-modules.

The **control network** bunch-by-bunch acquisition system for both monitors is based on VME PowerPC field stations. Each crate hosts a MVME7100 board running Linux with the Xenomai realtime extension and Tango control system software. One of the four Gbit Ethernet ports onboard is reserved for A|D|O and A|D|A communication. Upstream, at each machine shot, the AMCs acquire information from the field and transmit it to the control system via UDP/IP. A Xenomai real-time task running at the kernel level extracts from a dedicated Ethernet port, directly from the interrupt handler, the UDP data payload and performs further computation. A Tango server exports to the control system level the information collected by the real-time task. Downstream, a Tango server can interact with hardware; e.g. in the C-BPM case two slow loops regulate the amplitude of calibration signals and the gain level of analog front-ends.

BACK-END SYSTEMS COMMISSIONING

During the commissioning of the FEL, the diagnostics monitors based on the uTCA back-end were installed and put into operation. The following paragraphs will describe each specific application goal and achievements.

The BAM Case

Figure 7 shows a block diagram of a complete BAM station. In this setup, the back-end enclosed in a 1U uTCA crate consists of: one A|D|A, 1 Signal Conditioning Board (SCB) STM [1], and two 12 GHz photo-diodes.

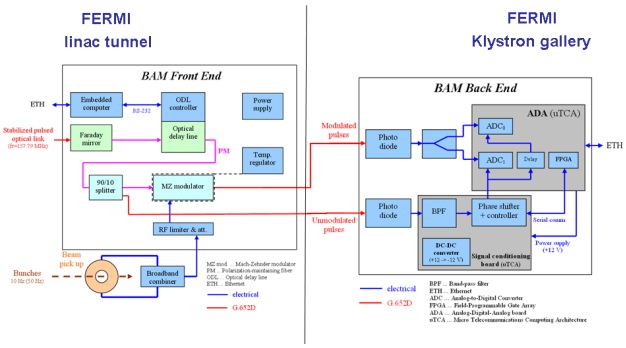


Figure 7: BAM system architecture.

The goal was to deliver three stations to characterize the FEL measuring its beam arrival time in strategic locations, with an accuracy of less than 20 fs. Major problems due to EMI interference inside the uTCA crate, forced designers to split the functionalities in separate crates. A custom chassis hosting the SCB and the Photodiodes had been therefore built. Installing two stations the goal of demonstrating the system principle was accomplished successfully, and an average system resolution of 7.66 fs was confirmed. Using BAM stations, measurements of the machine bunch compressor and some studies on the linac stability performance are now routine tasks of operators.

The C-BPM Case

In order to operate a C-BPM station the following modules are required: one A|D|A, one A|D|O, one MiTiCH, one OptioIO STM [2]. In this configuration, the back-end system enclosed in a 9U uTCA crate can host up to 3 C-BPM stations (Figure 8). The MiTiCH and one of the two A|D|A are shared between two monitors. The goal set was the delivery of 10 stations to operate the FEL with a beam position measurement of 1 um or better.

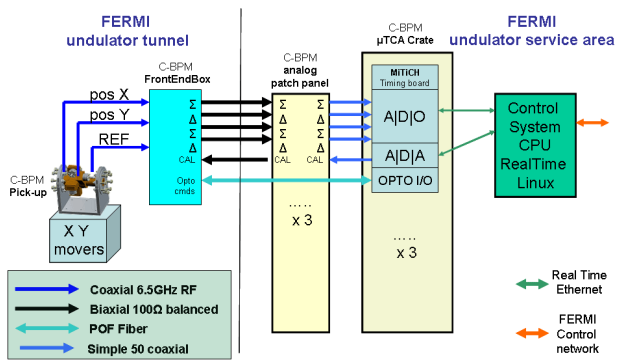


Figure 8: C-BPM system architecture.

EMI problems caused by the uTCA original 220 VAC PSU forced its replacement; no other issues were experienced in this case. All planned systems were installed and successfully commissioned delivering monitors with an average resolution of 1.7 um. Exhaustive documentation on system calibration and position calculation can be found in [9] and [10].

To meet the final resolution specification, further fine tuning of the frontend electronics is planned. The instruments had been of fundamental importance to transport the beam along the undulator chain to achieve FEL light in December 2010. Beam transport and fast trajectory feedback [11] are now control room routine operations based on these devices.

OTHER UTCA DEVELOPMENTS

For future system enhancement and to fulfil complete standard compliancy, other system features were tested. Figures 9-10 show a schematic view of two uTCA advanced functionalities:

1. Intelligent Platform Management Interface (IPMI)
2. Fast backplane connectivity to a CPU via PCI Express

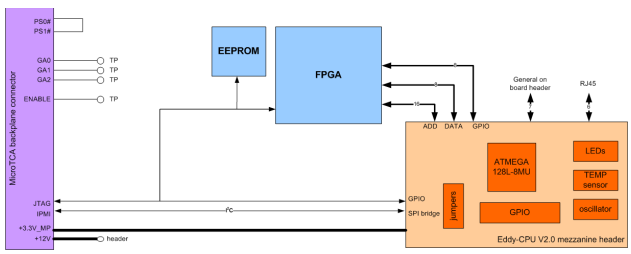


Figure 9: IPMI AMC controller .

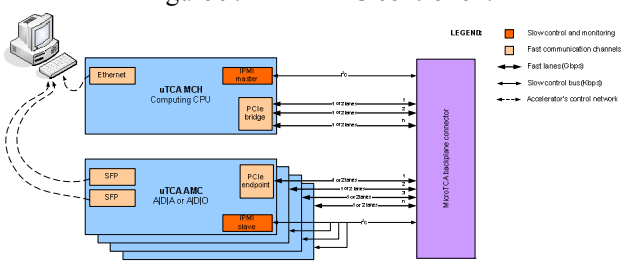


Figure 10: Fast AMC connectivity.

In collaboration with DESY in the framework of the IR-UV-X EuroFEL-PP, the above tasks were successfully carried out on an intensive measurement session at Elettra. (i) An Atmega 128L-8MU evaluation kit, wired to and A|D|O, proved the capability of the board to communicate to a commercial N.A.T. MCH for shelf management: hot swap and basic current and temperature readout and monitor were performed. (ii) A|D|O was successfully capable of transferring and receiving data via PCI Express to an ADLink CPU AMC1000 (Intel x86 64bit) and Sanblade SATA HDU, at a rate of 4 lanes at 2.5 Gbps (10 Gbps). The operating system and drivers used for the test were Linux based.

FIELD EXPERIENCE WITH UTCA

In general, the choice for uTCA as system architecture had been proved to be correct. It is a robust and reliable platform (5 machine runs without faults) handy to service and maintain. Especially working in the laboratory with its advanced functionalities, we had the chance to thoroughly explore the technology's potentials. EMI weakness in the crates evaluated were unexpected. Never the less the problems have been tackled and solved. We hope, especially in view of the physics extension (xTCA), that extra care on those issues will be taken by crate manufactures. We firmly believe that uTCA as is, will remain a valuable alternative for small scaled systems or modular systems were integration on AdvacedTCA (ATCA) blades is possible.

CONCLUSIONS AND OUTLOOK

Presently all systems required to operate the first FEL line (FEL1) of FERMI had been successfully installed and almost all commissioned. The total number of cards produced and tested sums up to 24 A|D|A, 18 A|D|O, and 4 MiTiCH. The number of uTCA crates on the field is 6. Those numbers will soon grow as the second FEL line will be commissioned. The platform and its modules are solid foundations for R&D in the digital electronics domain that we wish will continue also in the future.

ACKNOWLEDGMENTS

The author thanks all the Colleagues from the Timing and Diagnostics group at Elettra for the great professional collaboration and the pleasant social company. Special thanks to Mauro Pontremoli, TVR S.r.l, SCEN S.r.l, Officine Barnobi, and Piergiorgio Tosolini. Thanks to Patrick Gessler (and DESY) for the fruitful information exchange on uTCA related issues and developments.

REFERENCES

[1] L. Pavlovic et al., "Bunch Arrival Monitor at FERMI", BIW'10, Santa Fe, May 2010, TUPSM086

[2] R. De Monte et al., "the FERMI@Elettra Cavity BPM system: description and results", DIPAC'11, Hamburg, May 2011, MOOC03

[3] The MictoTCA specifications page MTCA.0 R1.0 http://www.picmg.org/v2internal/microTCA.htm

[4] Intelligent Platform Management Interface V 2.0 page http://www.intel.com/design/servers/ipmi/

[5] A. Borga et al., "The diagnostics' back-end system based on the in-house developed A|D|A and A|D|O boards", BIW 2010, Santa Fe, New Mexico USA

[6] A. Borga, "Back-end system and A|D|A module concepts for FERMI's diagnostics", ST/F-TN-09/14, FERMI@Elettra, November 2009

[7] A. Borga et al., "A|D|O module: how to improve space usage inside uTCA systems for C-BPM", FESD#014, FERMI@Elettra, August 2010

[8] A. Borga et al., "MiTiCH: a timing synthesis and distribution uTCA MCH", FERMI@Elettra, FESD#077, January 2011

[9] P. Craievich et al., "Cavity BPM design, simulation and testing for the FERMI@Elettra project", FEL2010, Lund-Malmo, August 2010, THPC11

[10] P. Craievich et al., "Commissioning of the Cavity BPM for the FERMI@Elettra FEL project", DIPAC'11, Hamburg, May 2011, TUPD14

[11] G. Gaio et al., "Commissioning of the FERMI@Elettra Fast Trajectory Feedback", these proceedings

SOLID STATE DIRECT DRIVE RF LINAC: CONTROL SYSTEM

T. Kluge, M. Back, R. Fleck, U. Hagen, O. Heid, M. Hergt, T. Hughes, R. Irsigler, J. Sirtl
Siemens AG, Erlangen, Germany
H. Schroeder, ASTRUM IT GmbH, Erlangen, Germany

Abstract

Recently a Solid State Direct Drive® concept for RF linacs has been introduced [1]. This new approach integrates the RF source, comprised of multiple Silicon Carbide (SiC) solid state Rf-modules [2], directly onto the cavity. Such an approach introduces new challenges for the control of such machines namely the non-linear behavior of the solid state RF-modules and the direct coupling of the RF-modules onto the cavity. In this paper we discuss further results of the experimental program [3] [4] to integrate and control 64 RF-modules onto a lambda/4 cavity. The next stage of experiments aims on gaining better feed forward control of the system and on detailed system identification. For this purpose a digital control board comprising of a Virtex 6 FPGA, high speed DACs/ADCs and trigger I/O is developed and integrated into the experiment and used to control the system. The design of the board is consequently digital aiming at direct processing of the signals. Power control within the cavity is achieved by an outphasing control of two groups of the RF-modules. This allows a power control without degredation of RF-module efficency.

MATERIALS AND METHODS

The system decomposition as shown in Figure 1 separates the system into two sub-systems, the control system and the cavity system (all block-diagrams are SysML [5] internal block diagrams (ibd)).

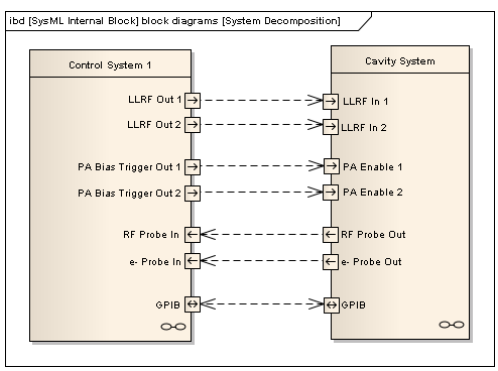


Figure 1: System Decomposition.

The cavity system (Figure 3) contains a 150MHz lambda/4 resonator that has a circumferential slot that separates the resonator into two conducting parts electrically isolated from each other. The current on the inner resonator wall is driven by a number of RF-modules each of which utilizing 8 SiC transistors in a parallel push-pull configuration for amplification of the drive signal. The power for the pulsed operation of the RF-modules is stored in a capacitor bank on the RF-module that is charged by a high voltage power supply unit (HVPSU). The charging voltage can be controlled by the control software. The drive signal for the transistors on the RF-modules comes from a pre-amplifier stage (PA) that amplifies the low level RF (LLRF) signal, splits and phase-shifts the amplified signal and adds a bias voltage to each output thus providing two ($\pm$180) drive signals for each RF-module. The operation point of each PA has to be set with a trigger pulse that starts before the LLRF arrives and stops after it. A calibrated antenna measures the electrical field in the cavity and is also able to detect currents caused by field emission, multipacting, ionization effects and field break-downs. We grouped the RF-modules into two groups allowing individual LLRF signals for each RF-module group (for reasons described later).

The first stage of experiments focused on demonstrating power combining and stable operation of the RF-modules on the cavity. Results with 32 RF-modules have been published in [3] and [4]. Since then the experiment has been moved to a new bunker and upgraded to run with 64 RF-modules in order to achieve higher electrical fileds within the cavity.

The analog control system for driving these experiments is depicted in Figure 2. It uses an Agilent N5181A MXG Analog Signal Generator to provide an RF-pulse with a certain frequency and amplitude set by the control software. The length of the RF-pulse is controlled by a trigger. The signal is split up in order to provide both (identical) LLRF output signals. The NI-Rack is a National Instruments (NI) PXI-1042Q chassis that hosts the control computer (NI PXI-8110 controller) and the timing system (NI PXI-7851R FPGA and NI PXI-6653 timing module). The timing system provides a clock signal and freely programmable triggers with 25ns resolution. The two oscilloscope are LeCroy Wave Runner 104MXi-A. Scope1 is software controlled and serves as a data acquisition device, the LLRF signal and the probe signals can be read by the control software. Scope2 independently of the control system software shows control signals from the system. The custom made breakout box optically decouples the NI rack from the experiment.

Figure 5 illustrates the control software concept. A LabVIEW [6] program with a graphical user interface (GUI) has been used for controlling the experiment. It takes care of all configuration and persistent data storage issues. Interactive operation of all software controlled devices can be achieved by device specific dialogs. The configuration scheme of the program allows it to easily add extensions (applications) that perform automated tasks.

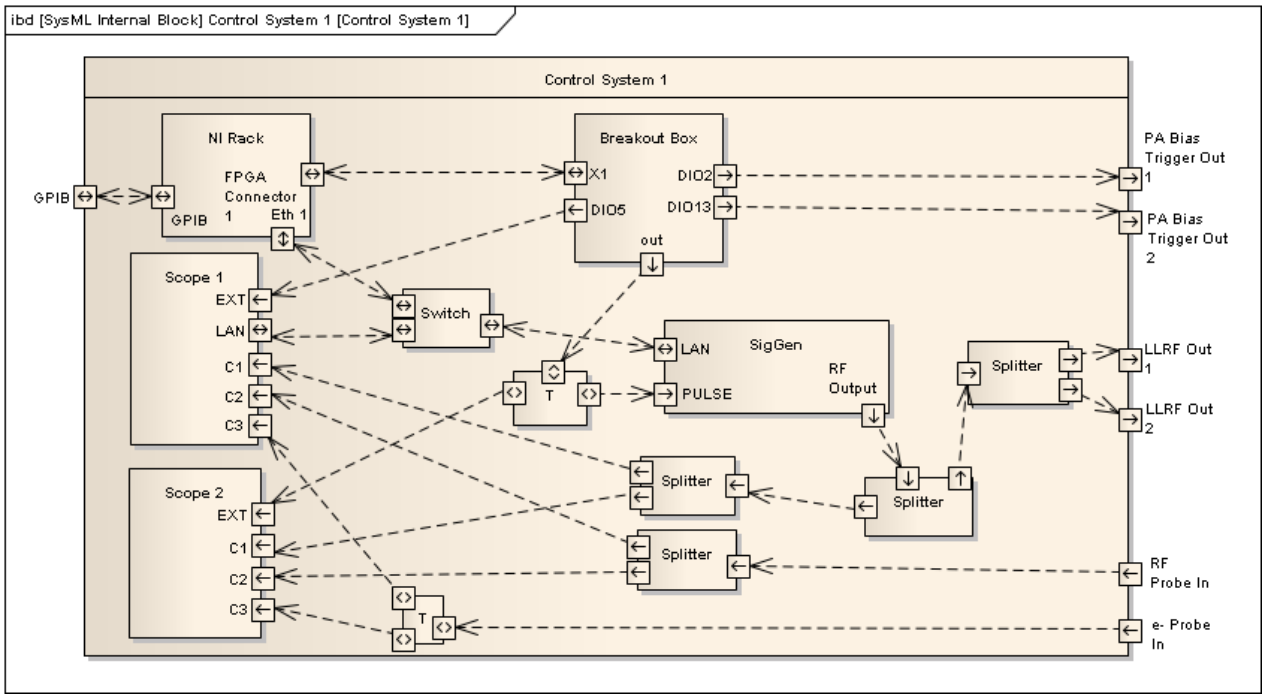


Figure 2: Analog Control System.

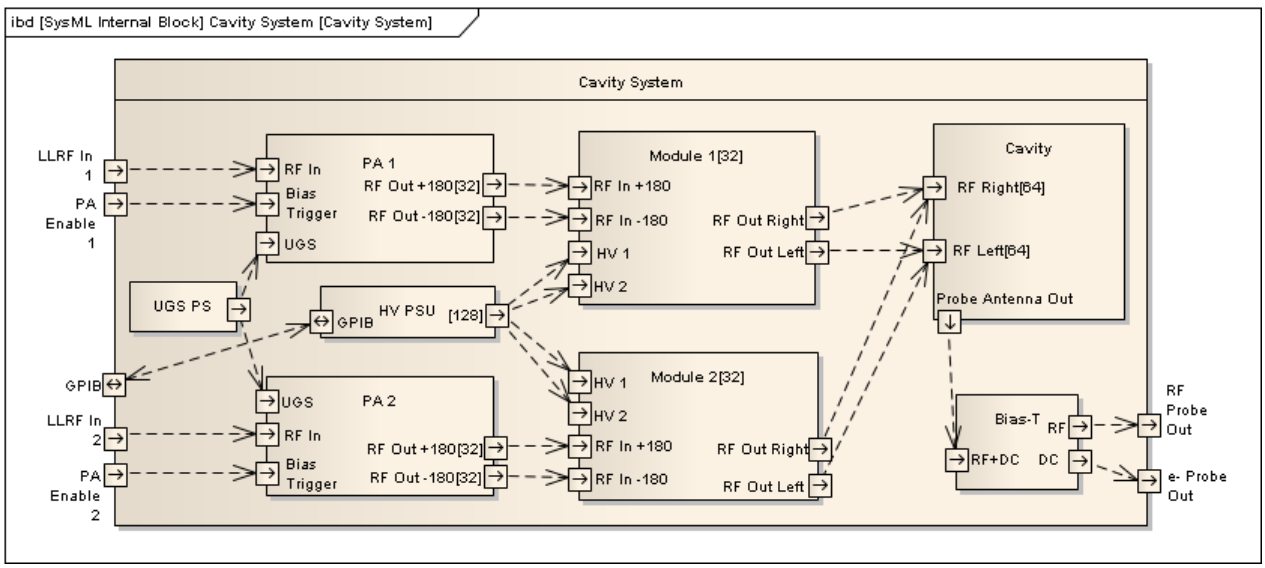


Figure 3: Cavity System (to control two groups of RF-modules).

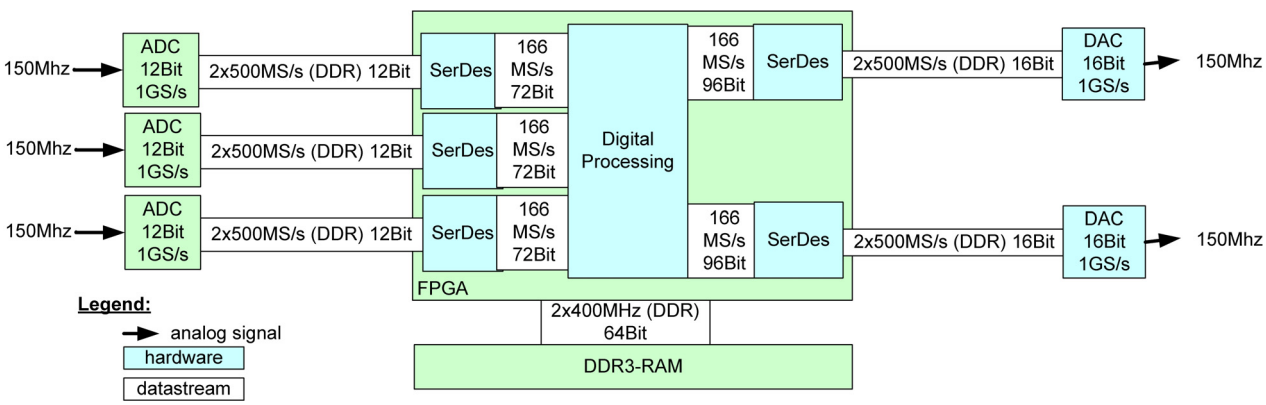


Figure 4: Data streams in the SCCRT.

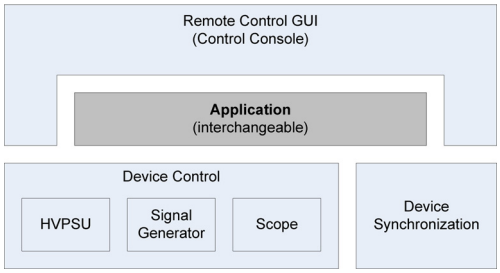


Figure 5: Control Software Concept.

Operating the directly driven cavity within the contect of an accelerator requires the ability to create an E-field with pre-defined and stable amplitude, frequency and phase during the whole particle injection phase of a macro-pulse. The analog control system used for the RF-module integration does neither have the control capabilities nor the timing accuracy required to achieve this.

The semiconductor driven RF-modules are non-linear time variant devices. In particular, the output power decreases over time due to discharging of the capacitor banks and self-heating effects of the SiC transistors during the pulse. The SiC RF-modules have their highest efficiency at maximum output power. In order to preserve the high RF-module efficiency while controlling the power fed into the cavity we are aiming at an outphasing control [7] like illustrated in Figure 6. Two groups of RF-Modules are driven separately with phases φ_1 and φ_2 and maximum output amplitude V_0. The cavity gap works like a combiner and inserts only the vector-summation V_{res} and φ_0 where

$$V_{res} = V_0 \cos\left(\frac{\varphi_1 - \varphi_2}{2}\right) \quad (1)$$

and

$$\varphi_0 = \frac{\varphi_1 + \varphi_2}{2}. \quad (2)$$

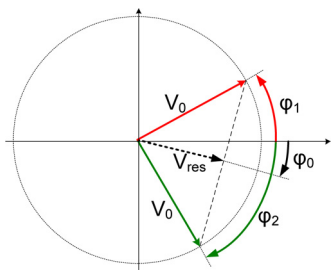


Figure 6: Outphasing Control Concept.

This requires that the control system can provide two LLRF signals that at least differ in their phases. We also would like to run measurements for detailed system identification. So we have to be able to acquire probe signals from the cavity and in the future also from the individual RF-module groups. We are expecting that these signals will have to be used in the future in order to realize a feedback control loop for increasing the accuracy of the cavity field. Therefore we decided to build a digital control system component called SCCRT (subsystem cavity controller real time) that gives us the required LLRF control and data acquisition capabilities and flexibility. The data streams in the SCCRT are illustrated in Figure 4. The core of the system is a Virtex 6 FPGA running at 166MHz. We have two RF output channels and three RF input channels for data acquisition. The 16 bit DACs and the 12 bit ADCs are running at 1GHz allowing the system to be used for RF frequencies up to 500MHz. The DDR RAM can be used for storing pre-defined RF-shapes and also for storing acquired RF and logging data.

The SCCRT furthermore can provide output trigger signals for synchronization of cavity system components. External events should be recorded and processed by the SCCRT as well, so the SCCRT also has input triggers. By handling all timing relevant issues within a single component we are aiming at maximum accuracy and minimum jitter. The SCCRT contains an internal 1GHz master clock.

The first revision of the SCCRT has recently been introduced in detail in [8].

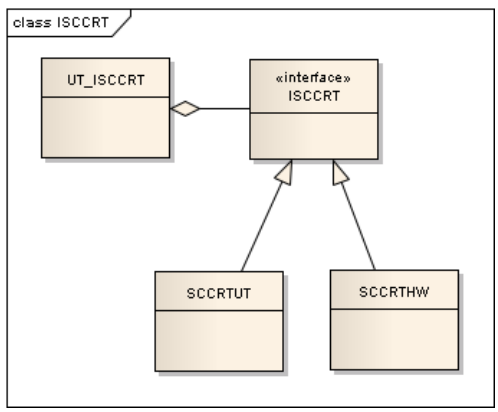


Figure 7: SCCRT Driver Class Diagram.

A LabVIEW [6] object oriented driver interface class (ISCCRT) has been used for defining the functionality and the programming interface for the first revision of the SCCRT, see Figure 7. The first driver client implemented is the class UT_SCCRT that provides a graphical user interface (GUI) for interactive tests of the driver. The class SCCRTHW provides the "real" driver communicating with the hardware while the class SCCRTUT is an implementation intended for validating that the unit test GUI is working correctly. Like intended, the use of the interface class allowed us to separate and parallelize the development of the software components. So the test GUI was available when the first hardware samples have been ready.

The next stage of experiments will start with the integration of the SCCRT into the control system replacing significant parts of the analog control system (see Figure 8).

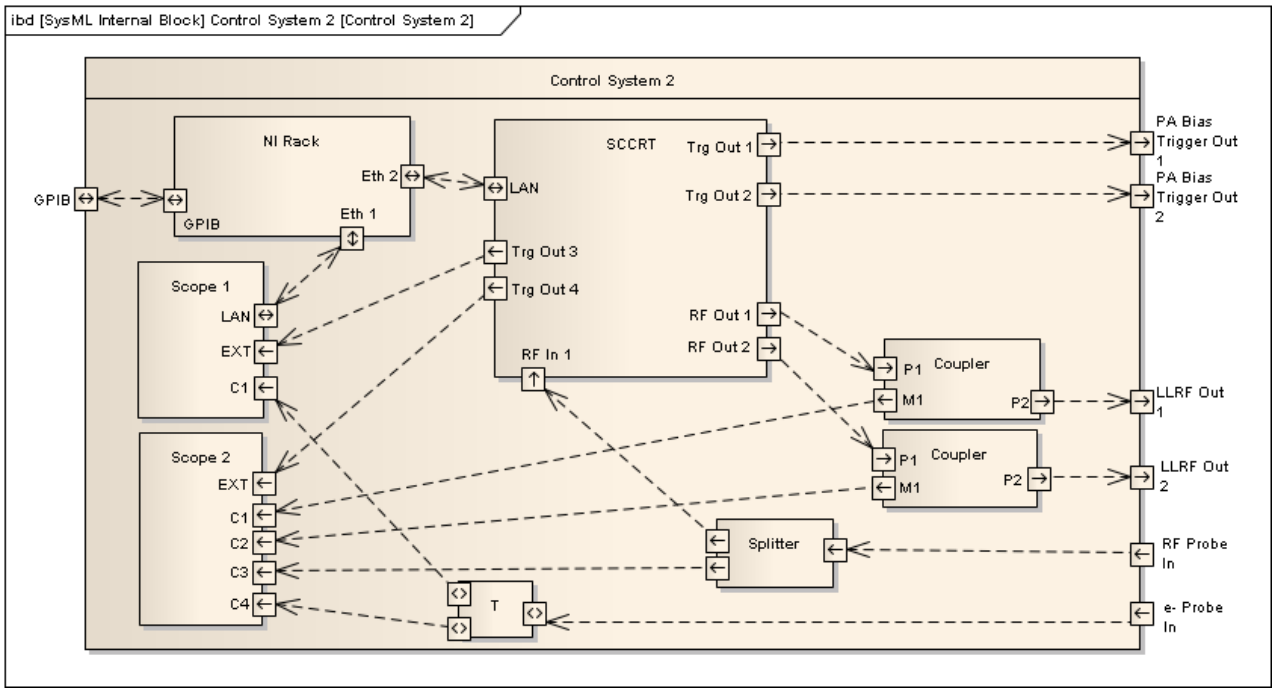


Figure 8: Digital Control System.

RESULTS

First commissioning tests with the 64 RF-modules have been carried out using the analog control system. At 150MHz and a RF-module supply voltage of 160V we have measured an E-field inside the cavity gap of approximately 66MV/m and a transmitted power of approximately 120kW.

The first revision of the SCCRT hardware has been assembled, the FPGA code is developed as well as the LabVIEW driver code. We have started the unit tests of the SCCRT.

NEXT STEPS

- Complete integration of the digital control system.
- The digital control system will be used for testing the outphasing amplitude control approach.
- The digital control system should be used for system identification purposes for gaining information for regulator design.
- The FPGA on the digital control system will be reprogrammed in order to support a closed loop operation with a PI regulator.
- Power injection tests with higher RF-module supply voltages should take us to the electrical field breakdown limits.

REFERENCES

[1] O. Heid, T. Hughes, THPD002, IPAC10, Kyoto, Japan

[2] Irsigler R. et al, 3B-9, PPC11, Chicago IL, USA

[3] O. Heid, T. Hughes, THP068, LINAC10, Tsukuba, Japan

[4] O. Heid, T. Hughes, MOPD42, HB2010, Morschach, Switzerland

[5] OMG Systems Modeling Language, `http://www.omgsysml.org`

[6] National Instruments LabVIEW, `http://www.ni.com/labview`

[7] H. Chireix, Proceedings of the Institute of Radio Engineers, Vol. 23, No. 11, p. 1370

[8] J. Sirtl et al, MOPC152, IPAC11, San Sebastian, Spain

ETHERBONE - A NETWORK LAYER FOR THE WISHBONE SoC BUS

M. Kreider, W. Terpstra, GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany
J. Lewis, J. Serrano, T. Włostowski, CERN, Geneva, Switzerland

Abstract

Today, there are several System on a Chip (SoC) bus systems. Typically, these buses are confined on-chip and rely on higher level components to communicate with the outside world. Taking these systems a step further, we see the possibility of extending the reach of the SoC bus to remote FPGAs or processors. This leads to the idea of the EtherBone (EB) core, which connects a Wishbone (WB) Ver. 4 Bus via a Gigabit Ethernet based network link to remote peripheral devices.

EB acts as a transparent interconnect module towards attached WB Bus devices. Address information and data from one or more WB bus cycles is preceded with a descriptive header and encapsulated in a UDP/IP packet. Because of this standard compliance, EB is able to traverse Wide Area Networks and is therefore not bound to a geographic location.

Due to the low level nature of the WB bus, EB provides a sound basis for remote hardware tools like a JTAG debugger, In-System-Programmer (ISP), boundary scan interface or logic analyser module. EB was developed in the scope of the WhiteRabbit Timing Project [1] (WR) at CERN and GSI/FAIR, which employs GigaBit Ethernet technology to communicate with memory mapped slave devices. WR will make use of EB as means to issue commands to its timing nodes and control connected accelerator hardware.

PURPOSE

EB is a network protocol meant for fast, low level software to hardware or hardware to hardware communication. It connects two distant WB SoC buses and is capable of direct memory access to attached devices. EB shall be used in GSI/FAIR and CERNs timing nodes as well as for remote debugging and programming, making hard to reach embedded systems easier to deploy and maintain.

RELATED WORK

There are many different examples of protocols for direct data exchange available. Among the most commonly used low-level were Myrinet in the supercomputing sector (almost completely replaced now by Ethernet based equipment) and different Remote Direct Memory Access (RDMA) [2] implementations. While there are pure software implementations of RDMA, their latencies cannot compete with hardware implementations like Infiniband [3] or iWARP [4], which can can achieve latencies below 7μs. However, these are mostly optimised for maximising throughput, while latency is still a secondary factor. There are also high level protocols available like CORBA [5] and SOAP [6], which aim for abstract software to software communication in heterogeneous environments. While being very versatile, due to their higher logistics overhead and generic nature, they are not well suited for fast communication between hardware and hardware or hardware - software. All of the above have in common that they are not tied to a specific underlying bus protocol of their endpoints. While they of course keep data content, they will not preserve syntax during transport.

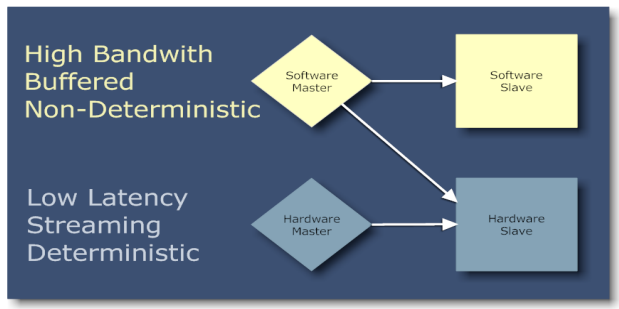


Figure 1: Compatibility between EB node types.

ARCHITECTURE

General Considerations

Since bus protocols can differ quite strongly in their workings and packet layout, conversion between them can severely reduce fidelity. For EB, we therefore chose Wishbone V4 as a concrete bus implementation, while leaving the underlying transport protocol open. There are two categories of EB devices under development: Buffered, non deterministic software modules and low-latency, deterministic streaming hardware cores (Fig. 1).

Software nodes are used for all applications where determinism and latency are not the main issue, but interoperability and fidelity of bus signals are. One example would be a developer's computer, remotely connected to the JTAG module of an embedded system elsewhere on the premises. Figure 2 shows an example block diagram of such a setup.

Hardware nodes operate in full streaming mode, They are fully deterministic and made to cut latency down to the minimum. An application example would be an endpoint of a timing system, receiving commands to generate a pulse at a specific execution time. The deterministic characteristics of EB ensure that available time frame for delivery does not vary, while streaming provides low latencies, reducing the time the control system needs to buffer. Hardware implementations are of course not as flexible as software.

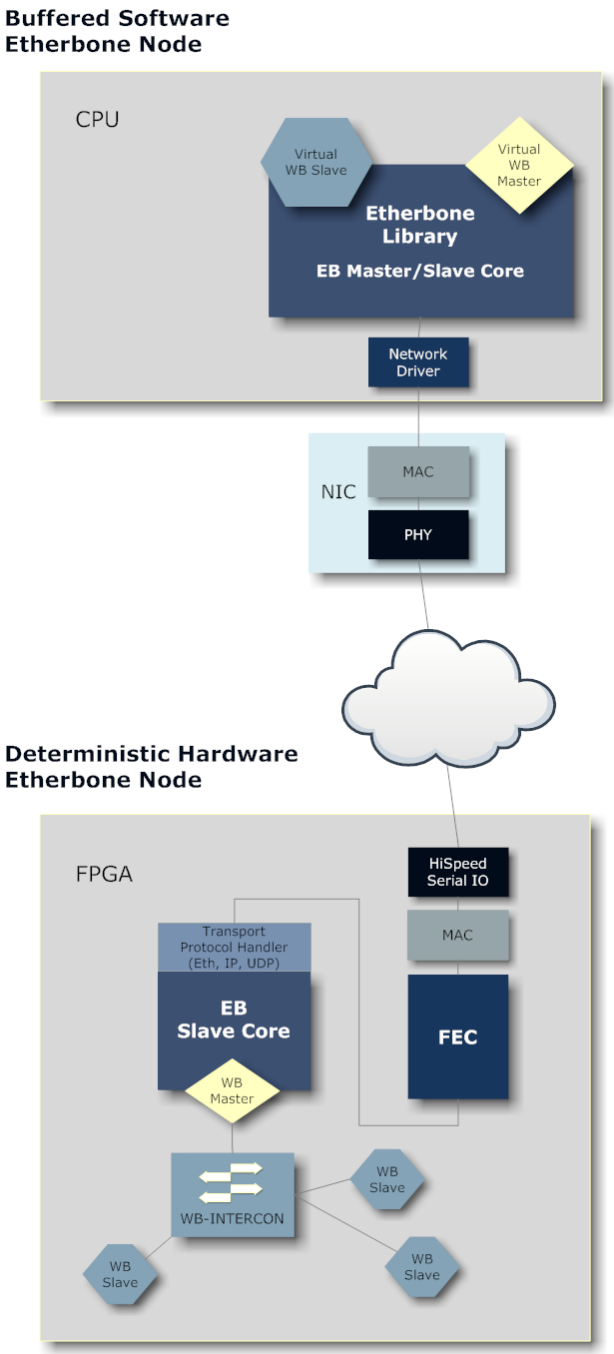


Figure 2: EB Slave node.

Methods and Test Implementation

Our current SW/HW test implementation utilises UDP/IP as a transport protocol. As a requirement, EB needs duplicate free transmission, which UDP alone cannot guarantee. It is therefore assisted by a Forward Error Correction (FEC) Scheme on OSI layer II. In order to be fully deterministic and to achieve lowest latency possible, EB needed to be fully streaming capable, ideally introducing no additional delay to passing data. A hard timeout is enforced when waiting for replies to EB requests, depending on the given time frame. The main problem is in the IP and UDP packet headers, which contain length information and checksums on the payload. On the Ethernet layer for example, the Cyclic Redundancy Check (CRC) follows the payload, so on the fly processing or insertion is not a problem. This is not the case with UDP/IP, where prior knowledge about the payload is necessary. For low latency streaming, we had to resolve the dependencies between packet header and payload. An example for a concrete implementation can be found in Figure 3, showing the internal layout of the Hardware Description Language (HDL) of a deterministic EB slave node.

Like the underlying WB bus, EB has master and slave nodes [7], which form complementary pairs. This leads to a bridge architecture, where an EB master accepts bus operations from local WB masters for transfer to a remote node. EB slaves therefore have a WB master interface and form the remote representation of the local WB master.

TRANSPORT PROTOCOL

The EB protocol has been designed to be deterministic with a focus on minimal latency. It also needed to be able to use standard transport layer architectures. Since WR utilises Ethernet technology, making EB interoparable with GigaBit Ethernet standard was an obvious choice in the development. Raw Ethernet frames however were not an option, because they cannot pass routers and firewalls without special configuration. So a widely supported protocol with very low overhead was needed, and the choice fell to UDP [8].

Packet Length

The UDP/IP Header requires packet length fields before the payload. To avoid waiting for packet completion, streaming EB replies are the same length as the corresponding request, making the full header knowable in advance. EB counters every incoming read operation by an outgoing write operation, while incoming Writes are answered with zero padding. Without any need for exceptions to the rules, these are treated as empty EB records, the result is similar to a no-operation instruction in a CPU.

Checksums

The IP checksum is only dependent on fields of the IP header. This includes source and destination address, IP packet options and the packet length field. When replying to a request, almost all information required can be taken from the incoming packet header, except for length fields, source IP address, IP checksum and UDP port. Source address and port are already known to the node, which leaves the payload length and the checksum itself [9]. Due to the symmetry we introduced, length is also known in advance and the IP checksum of the reply can be already be calculated after header reception. With this, all reply header information is available at the beginning of the incoming

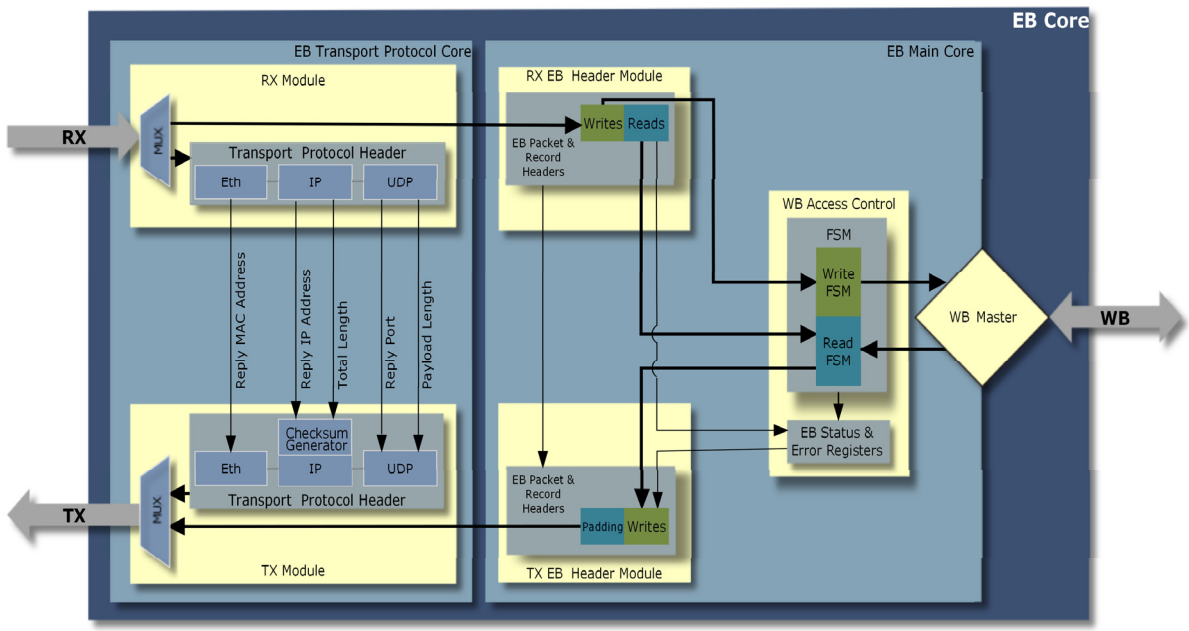


Figure 3: EB Slave node.

payload. This eliminates all wait times for payload reception, trading bandwidth for latency. The UDP Checksum is purely dependent on payload, but UDP protocol specifications allow the checksum to be set to zero. This signals the recipient "not used" [8]. Since we use a more powerful FEC in addition to the CRC, the UDP checksum can be omitted without risking data integrity.

ETHERBONE STRUCTURES

Packet Layout

A full EB Datagram is shown in figure 4. It consists of a header block and one or more record headers with matching Write or Read operations.

The header block starts with a magic word and two bit masks, used to signal the used WB bus width and address size. There is also the option to set a probe flag, which is used for negotiation of usable bus and address widths between two devices. A Probe-packet is always padded to the maximum alignment, 64 bit in our test case to ensure compatibility.

A record header contains a set of flag bits, stating optional information about source and destination, like the use of FIFO mode, etc. The flags are followed by the number of write and read operations in the record, this can be a number between zero and 255 if no feedback is necessary (assuming sufficient free space in the packet). If the bus is wider than the record header, it will be padded.

After the record header follow bus operations, Writes first, then Reads. Bus Operations are not mandatory, an EB record can therefore be empty, contain Writes, Reads, or both. Each of these blocks is preceded by an address field. For a Write, this field signifies the target start address on the slave; for a Read it is the address to which the read values shall be written to on the master.

Communication

Negotiation A typical EB connection starts by sending a probe packet to a slave. It solely consists of the header block with all possible bus and address sizes the master supports and a set probe flag. The slave then sends back the intersection of the offered modes and the one it supports itself. The result shows all possible bus and address widths the EB master can choose from for communication with this particular slave device, completing the negotiation. In the next step, a normal EB packet is sent, containing one or more EB records. The slave will reply with the bus width and address size chosen by the master in the request header.

Atomics EB supports atomic WB bus operations. While the cycle line is held, so is the connection to the target slave through a WB interconnect. Each Etherbone record comes with the option of ending the current bus cycle on completion or keeping it to the next record. With this mechanism, bus ownership can be held over several EB records, avoiding interference to the operation by other bus devices.

Symmetry Equal packet length of in- and outgoing traffic is essential for EB streaming mode. In order to keep equal length between request and reply, results from Reads are converted to write operations while Writes are turned

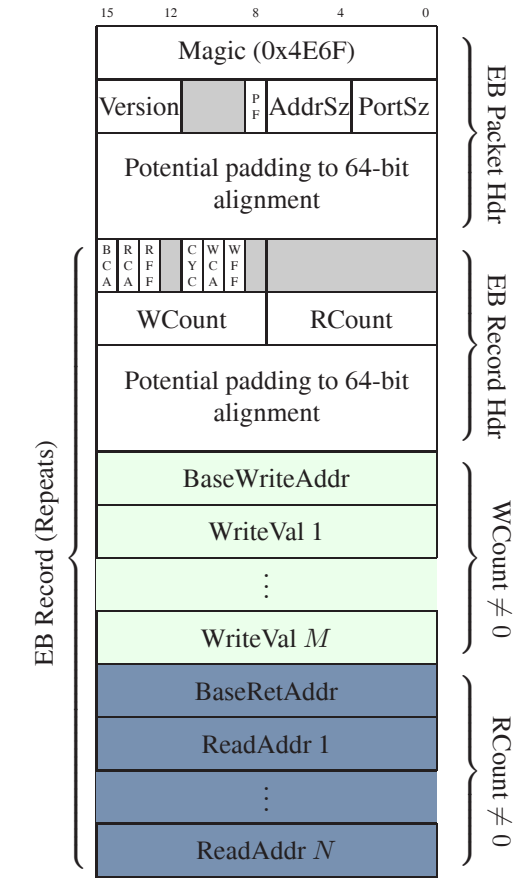


Figure 4: Etherbone v0.2 Message Format.

into empty Records, i.e. padding. This makes the length field of UDP/IP headers known in advance and resolves all header/payload dependencies.

Addressing EB Write operations are Values to be written, while Reads are addresses to be read. Write addresses therefore need to be generated by an increment to the start address. This increment can either be zero, in which case the target is treated as a FIFO, or match the byte alignment master and slave agreed on. This implies Writes to be sequential, while Reads can be random access. A WB master must keep track of read addresses in order to correctly interprete the answering Write.

Management EB also supports the use of a configuration space, an address space that is not associated with the local WB bus and only concerns the EB node itself. Here, settings like the MAC address or status information like the error log are kept. If the flag is set, all following operations will not be put on the bus but will point to the config space instead. If feedback for success of operations is required, a shift register in the config space can be read to supply the error bit for the last 32 operations on the WB bus interface.

CONCLUSION

Time driven HDL simulation and minor tests on evaluation boards for GigaBit Ethernet links show possible latencies below 1 μs. Further tests on final WR hardware later this year have yet to confirm these values, but intermediate results show EB as a well suited candidate for control systems and remote tools alike.

OUTLOOK

With open EB hardware and software implementations becoming available as open source projects, chances for it becoming a widely accepted WAN remote bus protocol within the timing and control systems community are increasing. Our next task will be full integration with GSI and CERN's next generation timing system. For GSI, this will happen on the example of a proton linear accelerator, the first component to be deployed in GSI's FAIR extension. Next steps will be the completion of a remote toolbox, containing an ISP and Debugger for use with our network capable embedded systems.

REFERENCES

[1] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, G. Gaderer, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet", IEEE Precision Clock Synchronization for Measurement, Control and Communication 2009, pp1-5, Brescia, October 2009

[2] A. Romanow, and S. Bailey, An Overview of RDMA over IP, Proceedings of the First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003), Feburary 2003

[3] J. Liu, J. Wu, D. K. Panda, "High Performance RDMA Based MPI Implementation over Infiniband", International Journal on parallel programming, Vol. 32, No. 3, pp167-198, 2003

[4] M. J. Rashti, A. Afsahi, "10-Gigabit iWarp Ethernet: Comparative Performance Analysis with Infiniband and Myrinet-10G", IEEE International Parallel and Distributed Processing Symposium, pp.290, 2007

[5] A. S. Gokhale, D. C. Schmidt, "Measuring and Optimizing CORBA Latency and Scalability over High-speed Networks", IEEE Transaction on Computers, Vol. 47, No. 4, 1998

[6] Y. Ying, Y. Huang, and D. Walker, "A performance evaluation of using SOAP with attachments for e-Science", Proceedings of UK eScience All Hands Meeting, pp. 796-803, 2005

[7] Opencores, "Wishbone B4 WISHBONE System-on-Chip (SoC)Interconnection Architecture for Portable IP Cores" (Standard), 2010 `http://cdn.opencores.org/downloads/wbspec_b4.pdf`, last visited 20.09.2011

[8] J. Postel, "User Datagram Protocol", RFC 768 (Standard), Internet Engineering Task Force, August 1980

[9] N. Alachiotis, S. A. Berger, and A. Stamatakis, Efficient PC-FPGA communication over Gigabit Ethernet, Proceedings of the International Conferences on Embedded Software and Systems (ICESS 10), pp. 1727-1734, Bradford, UK, 2010

SUB-NANOSECOND TIMING SYSTEM DESIGN AND DEVELOPMENT FOR LHAASO PROJECT*

Guanghua Gong#, Shaomin Chen, Qiang Du, Jianming Li, Yinong Liu
Department of Engineering Physics, Tsinghua University, Beijing, China
Huihai He, Institute of High Energy Physics, Beijing, China

Abstract

The Large High Altitude Air Shower Observatory (LHAASO) project is designed to trace galactic cosmic ray sources by approximately 10,000 different types of ground air shower detectors. Reconstruction of cosmic ray arrival directions requires a precision of synchronization down to sub-nanosecond, a novel design of the LHAASO timing system by means of packet-based frequency distribution and time synchronization over Ethernet is proposed. The White Rabbit Protocol (WR) is applied as the infrastructure of the timing system, which implements a distributed adaptive phase tracking technology based on Synchronous Ethernet to lock all local clocks, and a real time delay calibration method based on the Precision Time Protocol to keep all local time synchronized within a nanosecond. We also demonstrate the development and test status on prototype WR switches and nodes.

INTRODUCTION

Gamma ray source detection above 30TeV is an encouraging approach for finding galactic cosmic ray sources. All sky surveys for gamma ray sources using a wide field of view detector is essential for population accumulation for various types of sources above 100GeV.

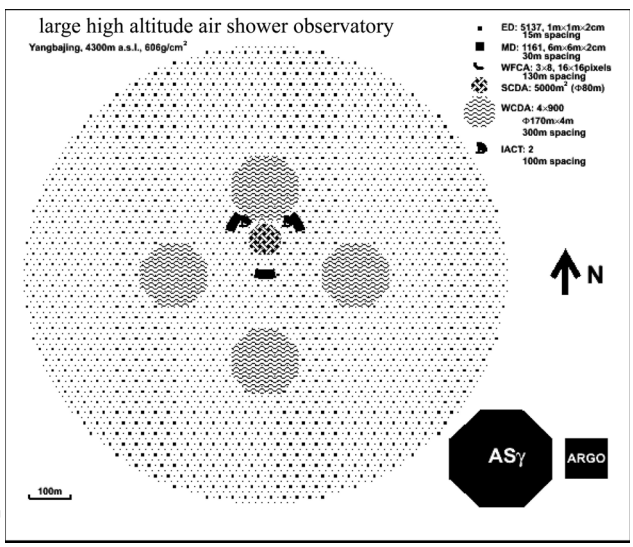


Figure 1: Layout of LHAASO observatory.

In order to target those objects, a large air shower particle detector array of 1Km^2 (the LHAASO project [1]) at 4,300 m a.s.l. is proposed, which is designed to cover a wide energy region from 30TeV to few EeV by combining together with many air shower detection techniques. The layout of the LHAASO observatory is shown in Fig. 1.

- KM2A: particle detector array with an effective area of 1km^2, including 5,137 scintillator detector (1m^2 each) to measure arrival direction and total energies of showers; 1,200 μ detectors to suppress hadronic shower background.
- WCDA: water Cherenkov detector array with a total active area of 90,000m^2 for gamma ray source surveys.
- WFCTA: 24 wide FOV Cherenkov telescope array
- SCDA: 5,000m^2 high threshold core-detector array.

To reconstruct the direction of cosmic rays, the arrival time difference of shower particles to each individual detector is measured using a uniform clock frequency. Differing from the classical structure which uses a central trigger as a common stop for measuring arrival time difference [2], a readout scheme without the central trigger is expected which means each individual detector will be self-triggered and the event must be precisely time tagged with a common time stamp reference.

Considering the pointing accuracy in reconstruction and the dimension of the detector array, a clock and time system is required to distribute time and clock to 10,000 nodes within 1 square Km area. Each node must have a local clock which is locked to a primary reference clock and have a local time with an accuracy of less than 1ns relative to the common reference.

There are also other considerations for this time and clock distribution system. Since it is neither practical nor realistic to manually calibrate the cable length of thousands of detectors, the system must have the capability to automatically calibrate and compensate the propagation delay caused by different cable length, temperature and mechanical stresses variation. The system should also be low cost, high reliable and low power consumptive.

EXISTING TECHNOLOGIES

Certain technologies exist for military, industry and commercial purposes with different features and performances:

- Radio/satellite navigation systems: like radio station, Global Positioning System (GPS), Global Navigations Satellite System (GLONASS), Galileo and Beidou.

*Work supported by Tsinghua University under collaboration with White Rabbit project (CERN) and National Science Foundation of China (No.11005065)
#ggh@tsinghua.edu.cn

- Wireless telecommunication: the commercial wireless telecommunications like CDMA, WCDMA, WiMAX/LTE also provide a time distribution.
- Physical Layer (Layer 1) synchronisation: like SONET/SDH, Gigabit Passive Optical Network (GPON), and Synchronous Ethernet.
- Packet-based synchronisation: like NTP, IEEE1588-2008 PTP, Circuit Emulation Services (CES) encapsulation and the future UTI J.211 from ITU-T.
- Ethernet based real-time network: like ProfiNet, EtherCAT, SERCOS III, INTERBUS and etc.

Method	Ability	Accuracy jitter	Medium	Layer	Complexity	Manageability
Radio Clock	Time	10ms	Wireless	Layer 1	Simple	No
NTP	Time	1ms	Wireless	Layer 3	Complex	No
CDMA	Time/Freq	10μs	Wireless	Layer 2	Complex	?
WCDMA	Time/Freq	3μs	Wireless	Layer 2	Complex	?
WiMAX/ LTE	Time/Freq	1μs	Wireless	Layer 2	Complex	?
GPS	Time/Freq	14ns	Sat – earth	Layer 1	Simple	No
PTPv2	Time	~ns	Ethernet	Layer 2	Complex	Yes
UTI J.211	Time/Freq	1ns	Cable	Layer 1	Simple	Yes
SDH/SyncE	Freq	10ps	Ethernet	Layer 1	Simple	No
White Rabbit	Time/Freq	<1ns	Fiber GBE	Layer 1, 2	Complex	Yes
Optical carrier sync	Freq	<50fs	Fiber	Layer 1	Ultra complex	Yes
Optical Frequency Comb distribution	Time/Freq	<10fs	Fiber	Layer 1	Ultra complex	Yes

Figure 2: Existing time distribution technologies.

Figure 2 lists the major features of several existing technologies. The PTPv2 and SDH/SyncE show the potential capability to be applied for the time and clock distribution in LHASSO project.

Facing the requirement of controlling thousands of nodes over few kilo-meters around large particle accelerate instruments, a novel technology, White Rabbit [3][4][5], has been developed in the frame of CERN's (and GSI's) renovation projects.

The White Rabbit technology shows a great applicability for LHAASO project:

- It can distribute clock and time with required accuracy
- It is possible to extend to 10,000 nodes.
- It has the feature of automatic link delay measurement and compensation.
- It has high reliability to support plug&play, no further tuning needed.
- It is an enhancement of existing data transmission links; does not need dedicated fiber or cable, which is cost effective.

WHITE RABBIT

White Rabbit is a combination and improvement of several technologies including synchronous Ethernet, precision time protocol, DMTD phase tracking and deterministic routing protocol.

Synchronous Ethernet

Synchronous Ethernet is a special extension of standard IEEE802.3 Ethernet standard where the recovered RX clock from its master is used as its own TX clock, making the whole system synchronous. With Synchronous Ethernet technology, the entire network can share a common clock to avoid any clock rate difference between nodes. This is done in hardware and is the basis for the PTP fine measurements.

Precision Time Protocol (IEEE1588)

The PTP [6][7] protocol synchronizes local time with the master time by measuring and compensating the delay introduced by the link. Link delay is measured by exchanging packets with precise hardware transmit & receipt timestamps which is explained in Fig. 3.

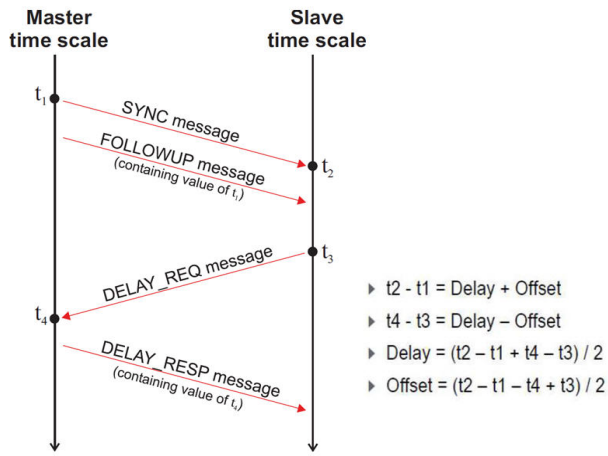


Figure 3: PTP timing process.

The Synchronous Ethernet technology provides the same clock frequency to the whole network which simplifies the PTP calculation that there is no clock rate difference but only time delay.

DMTD Phase Tracking

The measurement of standard PTP is limited by the clock frequency, a fine delay measurement of less than one clock cycle can be achieved by detecting the phase shift between transmit and receive clock on the master side.

A digital Dual Mixer Time Difference [8] (DMTD) clock measuring logic is shown in Fig. 4, which is easy to implement in a modern FPGA device.

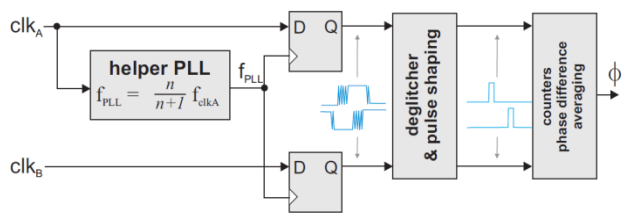


Figure 4: Digital DMTD.

DEMONSTRATION SETUP AND TEST

To evaluate the performance of Write Rabbit technology, a test system has been setup, including one WR switch and one receiver node, as shown in Fig. 5.

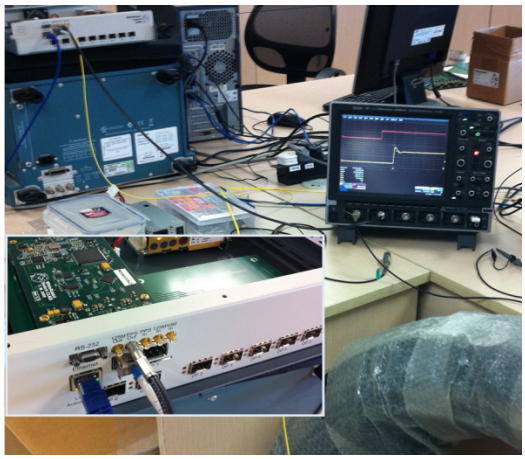


Figure 5: Demonstration setup.

The White Rabbit switch is a standalone device, the receive node is provided as an IP core and is tested on a PCI-E FMC carrier board, as shown in Fig. 6.

Figure 6: Slave node (SPEC).

The delay between the PPS of the switch and the receiver node is measured with a oscilloscope. The results are listed in table 1, which shows that the fiber length difference is transparently measured and compensated.

Table 1: Fiber Length Compensation Measurement

Fiber Length	PPS delay	
	Mean[1]	Sdev[2]
30cm	121.02ns	115.49ps
1km	125.72ns	110.66ps
5km	127.62ns	105.14ps

Note 1: the delay mainly comes from the length difference of the coaxial cables used for measurement.

Note 2: the deviation mainly comes from the test signal drive circuit.

The link was forced to re-establish to verify the stability of the time reconstruction. As presented in table 2, the peak-peak deviation among different runs is less than 100ps. The average shows a slight linear relation with the length of the fiber which is caused by the asymmetric delay factor of the 1550nm and 1390nm lights used for upstream and downstream respectively. This effect can be further reduced by calibrating and tuning a corresponding parameter in the delay calculation, which can be specific for certain optical fiber and transceiver.

WHITE RABBIT DEPLOYMENT IN LHAASO

Since the LHAASO project has around 10,000 nodes to be synchronized, a hierarchical structure of up to 3 levels must be implemented. The synchronization accuracy across multi-level switches is essential for this structure and a test for this purpose will be carried out soon.

Table 2: Repeatability of Recovered PPS (Unit: ns)

#Run	30cm	1km	2km	3km	4km	5km
Run 1	16.05	15.89	15.82	15.78	15.67	15.57
Run 2	16.05	15.92	15.89	15.76	15.64	15.65
Run 3	16.02	15.93	15.86	15.72	15.67	15.65
Average	16.04	15.91	15.86	15.75	15.66	15.62
Peak-Peak	0.03	0.04	0.07	0.06	0.03	0.08
Link delay	473	10305	20145	29969	39801	49641

A Rubidium/Caesium module provides a high accurate clock source as the external reference of the top switch, all the nodes will then trace this clock to recover a local clock; a high-precise GPS receive module provides a UTC time information as well as a Pulse per Second (PPS) signal to the top switch, all detector nodes will follow to re-establish a local time with an accuracy of less than 1ns.

Each type of detector in LHAASO project has its dedicated readout electronics to resolve the specific design issues. To save the cost and man-power of implementing the synchronized clock and time function among all detectors, a LHAASO wide clock and time mezzanine (CTM) will be developed to handle all the delicate details of white rabbit technology, each detector electronics will carry one CTM through a simple and straightforward interface.

The block diagram of CTM is shown in Fig. 7.

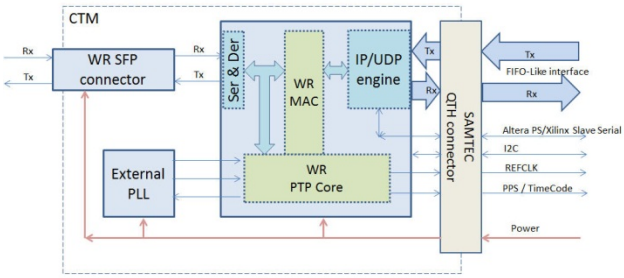


Figure 7: block diagram of CTM.

The CTM contains few major components:

- SFP connector: perform the O/E and E/O conversion.
- External PLL: to improve the precision of clock phase adjustment.
- FPGA: to implement the White Rabbit PTP core IP; it will also include a small IP packet process engine to simplify the interface to external logic as a FIFO. The engine can also provide a FPGA reconfiguration link for dynamic updating of external FPGA.
- Connector: a high-speed type from SAMTEC.

The recovered clock is directly feed to external logic, the time information will be encoded into a wide used format like IRIG-B such that the external logic will treat the CTM as a standard UTC device.

SUMMARY

The future LHAASO project requires high precise time and clock synchronization among thousands of detectors nodes. A novel design based on the White Rabbit protocol that could provide sub-nanosecond precision clock and time stamp distribution over the DAQ Ethernet fiber has been proposed. A preliminary test between a pair of master/slave nodes is presented and shows <0.4 ns rms time jitter between recovered master/slave PPS signals over different fiber lengths.

REFERENCES

[1] CAO Zhen, "A future project at Tibet: the large high altitude air shower observatory", Chinese Physics C, CPC(HEP & NP), 2010, 34(2): 249-252

[2] P.Allison, "Timing Calibration and Synchronization of Surface and Fluorescence Detectors of the Pierre Auger Observatory", 29th international Cosmic Ray Conference Pune (2005) 8,307-310.

[3] J.Serrano, "The White Rabbit Project", BE-CO Hardware and Timing section, CERN, Feb.7,2011

[4] J. Serrano, "THE WHITE RABBIT PROJECT", TUC4, ICALEPS2009, Kobe, Japan, Oct 2009

[5] Tomasz Wlostowski, "Precise time and frequency transfer in a White Rabbit network", master of science thesis, 2011, Warsaw University of Technology.

[6] "IEEE Std. 1588-2002 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" IEEE Std 1588-2002, pp. i–144, Nov. 2002, replaced by 61588-2004.

[7] J. Eidson, J. Mackay, G. M. Garner, V. Skendzic, "Provision of Precise Timing via IEEE 1588 Application Interfaces," in Proc. IEEE International Symposium on Precision Clock Synchronization for Measurement.

[8] G.Brida, "High resolution frequency stability measurement system", Review of Scientific instruments, Vol73, Number 5, May 2002

A REMOTE TRACING FACILITY FOR DISTRIBUTED SYSTEMS

F. Ehm, A. Dworak, CERN, Geneva, Switzerland

Abstract

Today, CERN's control system is built upon a large number of C++ and Java services producing log events. In such a largely distributed environment these log messages are essential for problem recognition and tracing. Tracing is therefore vital for operation as understanding an issue in a subsystem means analysing log events in an efficient and fast manner. At present 3150 device servers are deployed on 1600 diskless frontends and they send their log messages via the network to an in-house developed central server which, in turn, saves them to files. However, this solution is not able to provide several highly desired features and has performance limitations which led to the development of a new solution. The new distributed tracing facility fulfils these requirements by taking advantage of the Streaming Text Oriented Messaging Protocol (STOMP) and ActiveMQ as the transport layer. The system not only allows storing critical log events centrally in files or in a database but also allows other clients (e.g. graphical interfaces) to read the same events concurrently by using the provided Java API. Thanks to the ActiveMQ broker technology the system can easily be extended to clients implemented in other languages and it is highly scalable in terms of performance. Long running tests have shown that the system can handle up to 10.000messages/second.

INTRODUCTION

The Controls Middleware (CMW) project was initiated at CERN [1] in 2006 to provide a unified communication middleware for operating the particle accelerator infrastructure. This includes communication with servers that directly operate hardware sensors and actuators. It enables for example operators in the CERN Control Centre (CCC) to control equipment remotely via in-house developed Graphical User Interfaces (GUI).

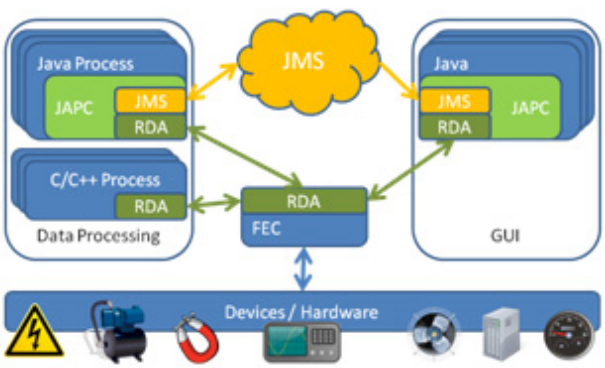


Figure 1: A simplified architectural overview on CERN's control system middleware components.

As illustrated in Figure 1 the CMW middleware is formed by a CORBA-based solution (RDA libraries) and a JMS based messaging infrastructure. Both cover similar use cases but they target at different system environments. JMS is used for high level middleware services and RDA for low level services running on hardware where very low message latency is demanded. The RDA library for example, is required by around 4000 Front End Servers (FES) running on 1600 Front End Computers (FEC) to allow getting/setting of hardware parameter values remotely.

Tracking potential issues in the middleware communication library layer is not an easy task as the involved services are distributed among many machines. In fact, there are two main problems:

- FECs are diskless systems running a real time LynxOS [2] operating system. Because there is no storage media it is not possible to write log messages to a local file.
- For other services which do have local disk storage it is difficult to correlate log events among them because files are distributed on various machines. Hence, accessing the data is a time consuming manual operation and slows down problem analysis.

Therefore, in the early days of CMW the idea of having a remote tracing facility was initiated and implemented. It allows collecting log events coming from RDA servers and from a subset of the middleware services.

THE CURRENT SYSTEM

In the current system log events are generated by the FECs and sent via UDP to a Java based central log server in plain text format. This server in turn converts the messages and writes them to a size-based rotating file.

Next to this, the CMW Admin GUI connects to the server and displays the log events received. It also allows setting a new log level via the RDA library remotely.

It is important to mention in this context that the services on FECs are hard real time processes which should be affected as little as possible by any other activity running on the same machine. Therefore, the communication between the CMW modules and the central log server is based on UDP as it is non-blocking.

Since the initial deployment this setup has been very useful for problem analysis and tracing. In particular the fact that a GUI enables a fast and easy information access makes it crucial for operation.

However, the demands have changed over time and the following problems have been identified:

- When setting a new log level in the CMW module all connected CMW Admin GUIs receive messages from the new level.

- Messages are lost due to a single threaded message acceptor in the central server.
- All messages go to one size based rotating file. When one source sends many events there is the risk that messages from other sources will be dropped.
- More messages need to be handled by the system since more services than originally planned are deployed.
- The C++ library only allows sending data to a remote host but not to a local file or to console output.

As a result of these problems it was decided to review and to implement a new central log event service.

REQUIREMENTS

As a first step towards a future solution the following requirements and limitations were defined:

- The new system must support log events from Java and C++ sources. Other languages may be supported.
- For C++ programs a lightweight library with little dependencies providing an easy API for storing events locally and sending them remotely should replace the old one.
- Operation overhead should be as low as possible and sufficient monitoring information must be provided.
- TCP for reliable and UDP for unreliable transport should be supported.
- Backward compatibility to legacy sources must be guaranteed.
- Support for precise timestamp information like microseconds and time zone where applicable.
- The log viewer must be provided as a standalone application, as well as a module which can be integrated into existing Java based diagnostic tools.
- Users should be able to read log messages remotely from a graphical user interface or via a console command *online* and *offline*. Online means that events are passed immediately to the client whereas offline means that the client has to actively poll for new events.

Restrictions

It should be mentioned that restrictions for the new system have also been defined:

- No operation critical system should depend on the distributed log event service.
- Messages may be lost due to inherent characteristics of the UDP transport layer.
- Only services running unattended are targeted to be used. This excludes highly dynamic user programs such as GUI's or console commands.

EXISTING MARKET SOLUTIONS

Collecting and storing log events from processes running on a Linux/Unix system is a well explored area and standardized for example via the widely used *syslog* protocol [3]. Here, processes send messages to a service running locally. The default for most Linux derivatives is *syslogd* which stores these events to local disk or forwards them to a remote syslog service via UDP. Other implementations like *rsyslogd*, *msyslogd* or *syslog-ng* exist and provide more features than syslogd. In addition, there are commercial and non-commercial tools to do analysis on the collected data. They allow an easier access via graphical interfaces (e.g. *Chainsaw* [4]) and partly provide extensive search and trending capabilities (e.g. *LogZilla* [5] and *Splunk* [6]).

Why Can We Not Take Existing Solutions?

As for the acquisition layer some of the present syslog implementations cover parts of the requirements but the main problem resides within the protocol itself. It contains limitations such as the missing support for microseconds and the maximum payload length of 2048 Bytes. Apart from this, none of the present syslog libraries supports setting the log level remotely

As for the C++ logging library there are many solutions available. However, most of them are in an early phase of development, do not support fully the requirements, are not stable enough or their dependencies are not supported on the current FECs.

Most of the existing GUI applications which are able to read events online are language dependent and connect directly to the log source. However, the new system requires reading log events from sources implemented in various languages and forbids direct connection.

Products like LogZilla or Splunk operate offline and provide fully featured tools for analysis and charting. However, they are proprietary solutions and do not allow to be integrated into existing in-house developed tools.

THE NEW ARCHITECTURE

Because there is no existing solution which satisfies the previously listed requirements a new central log event service has been implemented in Java and C++. As Figure 2 shows the *Converters* are responsible for accepting incoming log events, converting them into an internal common message format and sending them to the messaging service where they again can be read out by a *LogReader* or by a simple console reader.

Additionally, selected data may be stored permanently into a database or into local files.

The Log Sources

A program which requires sending data to the central service has two possibilities to do so: the first is the Linux standard syslog protocol and the second the newly developed *CMWlog*. Both can be used from C++ and Java. The main difference is that CMWlog supports microseconds and time zone information as well as TCP and UDP. It facilitates the STOMP [7] message frame and hence, is clear text based. STOMP is an open source protocol chosen for many messaging solutions [8] from various vendors and has gained great popularity in recent

years. Because it is so largely supported it was chosen for this project.

To allow Java based systems without the introduction of a large dependency set a new extension for the widely used Log4J [9] framework has been written.

The New C++ Log Library

CMWlog is a simple but full-featured logging library for C++ programs which - to a reasonable extent - resembles the Log4j Java library, providing flexible logging to files, to network, or to other destinations. It was developed to replace the old CMW logging library supports several platforms such as Linux and Windows but also old PowerPCs running LynxOS.

The library is fully configurable from the code or via configuration files. Like the previous version it supports setting the effective log level remotely. At the same time CMWlog is written to be thread-safe and it represents small memory footprint and resource consumption which is extremely important for previously mentioned old systems. Another important feature is the non-blocking method calls from the running user thread and proper behaviour in case of I/O errors not to affect time critical user's code execution.

The library utilizes concepts like *Loggers* and the *Appenders* [9]. These allow a high flexibility when channelling log events to different destinations such as standard output, files, or sending to a remote host. If required, the modular architecture of the library facilitates to be extended to other output destinations.

The Converters

A Converter is protocol-specific and hence accepts log messages only from particular remote sources. Internally, it is implemented using the Apache Camel [10] open source routing framework which features a very rich set of functionalities helping to reduce the development massively. It handles socket and thread operations and passes the received content to the user code. Here, the data is then turned into an internal common log message and subsequently sent off to the broker via TCP.

Currently, there are three converters. The first receives messages following the RFC5424 syslog format. The second reads messages coming from the new CMWLog log sources and the last one ensures backward compatibility with old legacy sources.

It is very simple to add new converters and thus to extend the usage to other log protocols. A concrete example for this is to accept also binary objects coming from Java programs using the Log4J or SL4J [11] log library.

To ease the operation monitoring metrics as well as management features such as blacklisting of log sources are exposed via the JMX [12] interface.

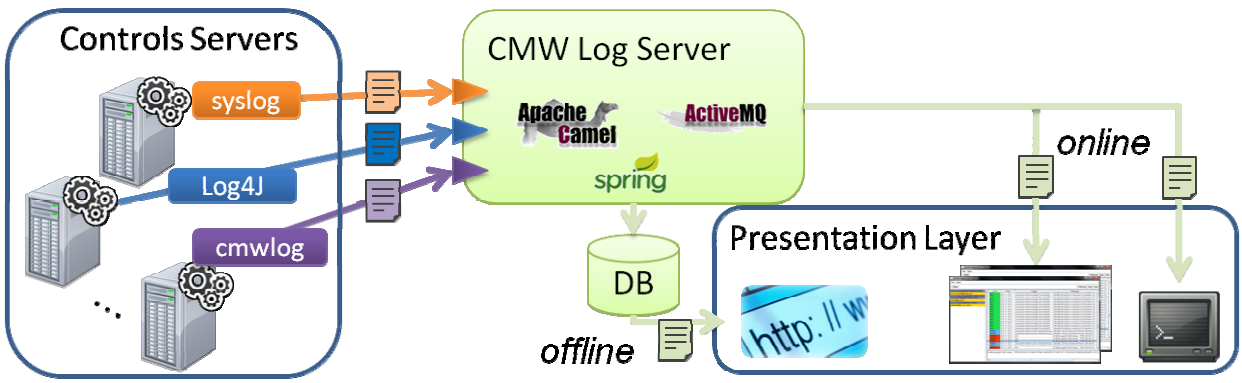


Figure 2: Architectural Overview of the new CMW tracing service.

Distribution of the Data

After a Converter have accepted and converted a log event it is then sent via JMS to a central messaging bus provided by the Apache ActiveMQ [13] product. ActiveMQ supports failover and clustering mechanisms, and has been used in the CERN controls system since 2005. As it has proven to be very reliable it was a natural choice for distributing log events to a potentially large set of client applications.

Data arriving at the broker is organized internally in topics. That is, each log source is identified globally by a unique identifier which maps in the broker to one specific topic. Clients may then read messages from one or more log sources by subscribing to one or more topics.

Storing Log Events Permanently

In order to fulfil the requirement of storing log events into a permanent storage two independent modules have been added to the system (see Figure 2). The *FileWriter* writes into size-based rotating files so disk space is not exceeded. At the same time the *DbWriter* pushes the data into a database (i.e Oracle or MySQL). In order to reduce storage usage, project specific policies on how long the data need to be kept have been put in place. For the CMW project, for example, data with warning and error severity are stored for one month only.

The Log Viewer

To finally view log events online and in real-time a graphical user interface (GUI) has been created. It displays the loggers in a tree structure and allows fine control of the desired levels. The messages are displayed in a table using a different colour for each severity.

The interface can run as a standalone application or as an integrated component within other GUIs. Moreover when connected to an RDA server it can remotely change the log level of the source easing on-the-fly problem tracing.

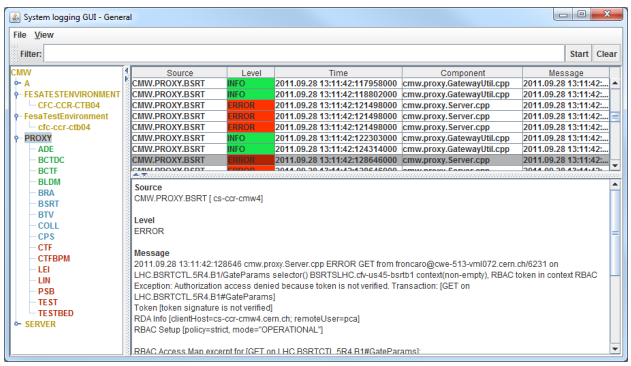


Figure 3: Example of the CMW log event viewer.

In addition, data can be viewed offline via an oracle driven web interface which provides sophisticated data filtering and selection tools. In case of an incident like a system or service crash this information can be very useful for later analysis. Another example is comparing trace messages produced by different sources at a given time with messages from another time. This gives the possibility to identify relationships between incidents.

PERFORMANCE TESTING

The new system has been deployed and tested in the very early stages of development. The focus was put onto the messaging broker as it handles most of the workload. For this, an ActiveMQ 5.4 message broker was installed on a 16 Core HP G6 server with 8GByte Memory. In order to simulate future workload two producers where set up on different machines sending in total 10.000 log events per second at constant rate via 1000 connections (TCP) to the broker. Each of these events contained a payload of 500 Bytes.

On the reading side 15 Java client applications where started each subscribing to a subset of the published data. Because one message is multiplied by the number of subscribers the effective outgoing message rate resulted in 30.000 messages per second. These performance numbers are slightly above the estimated future load to allow a better dimensioning of the system so bursts are handled safely, as well.

This test setup ran for 48 hours without any problems showing that ActiveMQ handles reliably the simulated load.

EXTENDING TO OTHER FIELDS

Because of its flexibility and simple use the new system has been investigated for transporting and storing important deployment and configuration feedback messages coming from kernel modules running on FECs. This feedback channel is currently missing but highly demanded. Developers have no means to verify the correct installation or reconfiguration of those modules.

Therefore, feedback messages were tested to be sent to the CMW tracing system and subsequently inserted into the database. This was successful and confirmed that the proposed system is a mature solution.

The same channel could possibly also be used for normal Java or C++ programs services. Information about the version, installation time and configuration could be collected centrally in a database for further use like viewing deployment history.

SUMMARY

The new tracing system fulfils all requirements and has been tested for performance and stability. It is designed to serve multiple reading applications for many equipment or service experts without putting additional load onto the log source and it offers to store the log events in a database or size-based rotating files. Users have the possibility to read messages online via a graphical user interface which was integrated into existing operation and diagnostic tools in the CERN control system. As data is also stored in a database the usage of existing market solutions for analysis is possible.

Because it allows being easily extended to other log protocols like Log4J or SNMP it offers an interesting solution to a larger field of activity.

REFERENCES

[1] CERN, www.cern.ch
[2] LynxOS, A real time operating system, http://www.lynuxworks.com
[3] The syslog standard, IETF, http://tools.ietf.org/html/rfc5424
[4] Chainsaw, Apache Log4J LogViewer, http://logging.apache.org
[5] LogZilla, http://www.logzilla.pro
[6] Splunk, http://www.splunk.com
[7] STOMP, Streaming Text Oriented Messaging Protocol, http://stomp.github.com
[8] Enterprise Messaging Solutions Technical Evaluation, Lionel Cons, Massimo Paladin, CERN
[9] Apache Log4J, A Log Library for Java programs, http://logging.apache.org/log4j
[10] Apache Camel, An enterprise routing framework, http://camel.apache.org
[11] SLF4J, Simple Logging Facade for Java, http://www.slf4j.org
[12] JMX, Java Management Extension, Oracle : http://www.oracle.com
[13] ActiveMQ: http://activemq.apache.org

COORDINATING SIMULTANEOUS INSTRUMENTS AT THE ADVANCED TECHNOLOGY SOLAR TELESCOPE

S. Wampler, B. Goodrich, E. Johansson, National Solar Observatory, Tucson, AZ, U.S.A

Abstract

A key component of the Advanced Technology Solar Telescope control system design is the efficient support of multiple simultaneously operating instruments sharing the light path provided by the telescope optics. The set of active instruments varies with each experiment and possibly with each observation within an experiment. The flow of control for a typical experiment is traced through the control system to present the key aspects of the design that facilitate this behavior. Special attention is paid to the role of ATST's Common Services Framework in assisting the coordination of instruments with each other and with telescope motion.

INTRODUCTION

The Advanced Technology Solar Telescope (ATST) [1] is being constructed atop Haleakalā on the island of Maui, Hawaii. With a 4m off-axis primary mirror, ATST is the world's largest solar telescope and includes fully integrated adaptive optics, an unvignetted light path, a 16m rotating Coudé lab and numerous other novel features. ATST is designed to be operated as a laboratory with multiple instruments sharing the light path. Each instrument includes one or more science cameras, each delivering data to the ATST Data Handling System (DHS) at up to 1GBps [2]. During observations, the active set of instruments must cooperate and coordinate with telescope motion and light-path adjustments.

The ATST control system manages this cooperation and coordination using a pair of principal systems: the Observatory Control System (OCS) and the Instrument Control System (ICS). The ICS manages the coordination of the individual instruments while the OCS coordinates the Telescope Control System (TCS) actions with the ICS. This separation allows the OCS to focus on large-scale coordination of an entire experiment (for example, instructing the ICS to prepare instruments while the TCS is active) without having to deal with the additional complexity of managing a cooperating set of instruments. Meanwhile, the ICS focuses on the coordination of the active instruments within an individual observation without having to deal with the coordination of telescope motions with those instruments.

All of the ATST *principal* software systems (OCS, ICS, TCS, and DHS) are built using the ATST Common Services Framework (CSF) [3] which provides significant support for their operation in a highly distributed environment. CSF is built on a container/component model with containers providing essential services and lifecycle management to application components. Components are the fundamental building block for all ATST applications. A special form of component, the *controller*, implements a command/action/response model for device control [4] where configurations describing changes of state are submitted to controllers. Controllers then act upon these configurations and, sometime later, report their success or failure in achieving the target state. Services are provided based as much as possible on need and not implementation. For example, CSF provides a middleware-neutral interface for communications among components. CSF also provides substantial *technical architecture* support (how something must be implemented to integrate into ATST software), allowing software developers to concentrate more on the *functional behavior* of their code (what needs to be done to accomplish a task). The topmost layer of CSF, *Base* [5], maps the generic service framework provided by the lower levels of CSF into a technical architecture more tailored towards the actions performed in observatory operation. Base provides motion controllers, hardware connection services, and numerous other pieces commonly needed when controlling mechanical devices. Often a software developer need only adjust an existing motion controller's parameters to produce a specific mechanism control application. Typically, this simply requires adjusting some property values in a database.

EXPERIMENTS

Observing with ATST is organized into *Experiments*. An Experiment is a formal description of the behavior of ATST required when carrying out the activities associated with a scientific proposal. The Experiment includes a description of the science goal, the conditions that must be met to perform the science, what instruments are required to obtain the data, and a *Program* describing how the data is to be collected using those instruments. The OCS executes the Program to produce *Observing Blocks*. Observing Blocks are the smallest schedulable section of a Program. Each Observing Block contains a list of the instruments that are active when the block is executed, a script that coordinates the TCS and ICS actions during that execution and parameters for configuring the ICS, TCS, DHS, and the instruments for proper behavior. When the Observing Block is to be executed, the OCS runs the script using a CSF-provided script executor pool and passes all of the parameters to the script. The use of a script to manage the coordination of the TCS and ICS maximizes the flexibility available within ATST's laboratory structure. Because the script executor resides in an ATST Java container, the scripts have full access to the facilities provided by CSF, easing the development effort.

Figure 1 shows a simplified (monitoring code, checks for external cancelation, and most comments have been stripped) script used when executing an Observing Block that performs a mosaic observation around some target

```
from atst.ocs.util import Mosaics  # Other imports provided automatically

tcs = App.connect("atst.tcs")   # Establish connections to TCS and ICS
ics = App.connect("atst.ics")

# Get the list of relative positions for the specific type of mosaic
pos = Mosaics.mosaicFactory(paramSet)

# Get the current target position on the Sun.
curLoc = tcs.get(Misc.setTarget("atst.tcs", 0.0, 0.0))
target = curLoc.getDoubleArray("atst.tcs.target")
oldx = target[0]
oldy = target[1]

mosaicType = paramSet.getString("atst.ocs.script:mosaicType")
Log.note(mosaicType+' mosaic start about ('+`oldx`+','+`oldy`+'):'+
                    paramSet.getId())
icsAction2 = None
for i in range(0, len(pos)):  # Step through mosaic positions
    x = oldx+pos[i][0]        # Get absolute position
    y = oldy+pos[i][1]

    psIcs1 = paramSet.selectOnPrefix("atst.ics.")
    psIcs1.insert(Attribute("atst.ics.tcsConfigured","false"))
    icsAction1 = submit(ics, psIcs1) # Tell ICS to configure instruments

    # If not first time through the loop, must block here until previous
    #    ICS action completes before moving telescope!
    if icsAction2 is not None:
        icsAction2.waitForDone()

    # Move the telescope and block until in position.
    psTcs1 = paramSet.selectOnPrefix("atst.tcs.")
    psTcs1.merge(Misc.setTarget("atst.tcs", x, y))
    submitAndWait(tcs, psTcs1)

    # TCS is now ready, tell ICS it's ok for instruments to collect data
    #   (makeConfigROE() adds the necessary parts to the configuration.)
    psIcs2 = makeConfigROE(psIcs1)
    icsAction2 = submit(ics, psIcs2)  # More efficient to delay blocking

# Out of loop, block here until last ICS action completes
if icsAction2 is not None:
    ics2.waitForDone()

tcs.set(curLoc)             # Return to starting point
Log.note(mosaicType+' mosaic done about ('+`oldx`+','+`oldy`+')')

App.disconnect("atst.tcs")  # Close connections
App.disconnect("atst.ics")
```

Figure 1: An OCS mosaic script (Jython code).

feature on the Sun. The script interleaves and coordinates instrument actions with telescope motions for efficient observing. Note that the script does not worry about the individual instruments – that is left to the ICS. Similarly the ICS does not need to worry about telescope motions – the OCS script tells the ICS when the TCS is properly configured and when it is not.

INSTRUMENT CONTROL SYSTEM

The Instrument Control System is the principal system responsible for the simultaneous operation of the ATST facility instruments. It is a layer of software between the OCS and the instruments that implements a simple control interface for the instruments and relieves the OCS of the burden of managing them. The ICS responds to OCS commands and events, coordinating and distributing them to the various instruments as required. The ICS presents a single narrow interface to the OCS while coordinating the actions of the instrument controllers underneath it.

The ICS requires no specific knowledge about the instruments to be controlled. All information about the instruments used in an experiment and their parameter settings is passed by the OCS through the ICS, which uses a list of participating instruments to extract and forward the parameters to the appropriate instrument controllers (ICs). All ICs, which do have detailed knowledge about their underlying instruments, use the same standard narrow interface. This allows new instruments to be added without having to modify the interface or any existing ICs. If an instrument implements the ATST Facility Instrument Interface it can be controlled by the ICS. This is true for both facility instruments as well as visitor instruments.

The ICS consists of two main components: the Observation Management System (OMS) which provides the interface to the OCS, and a set of Instrument Adapters (IAs) which provide a standardized interface between the OMS and their respective Instrument Controllers. A context diagram illustrating the structure of the ICS is shown below in Figure 2.

The interface between the OCS and ICS is configuration-based and essentially stateless. Commands from the OCS are directed to the ICS using CSF configurations. The configurations are used to send the

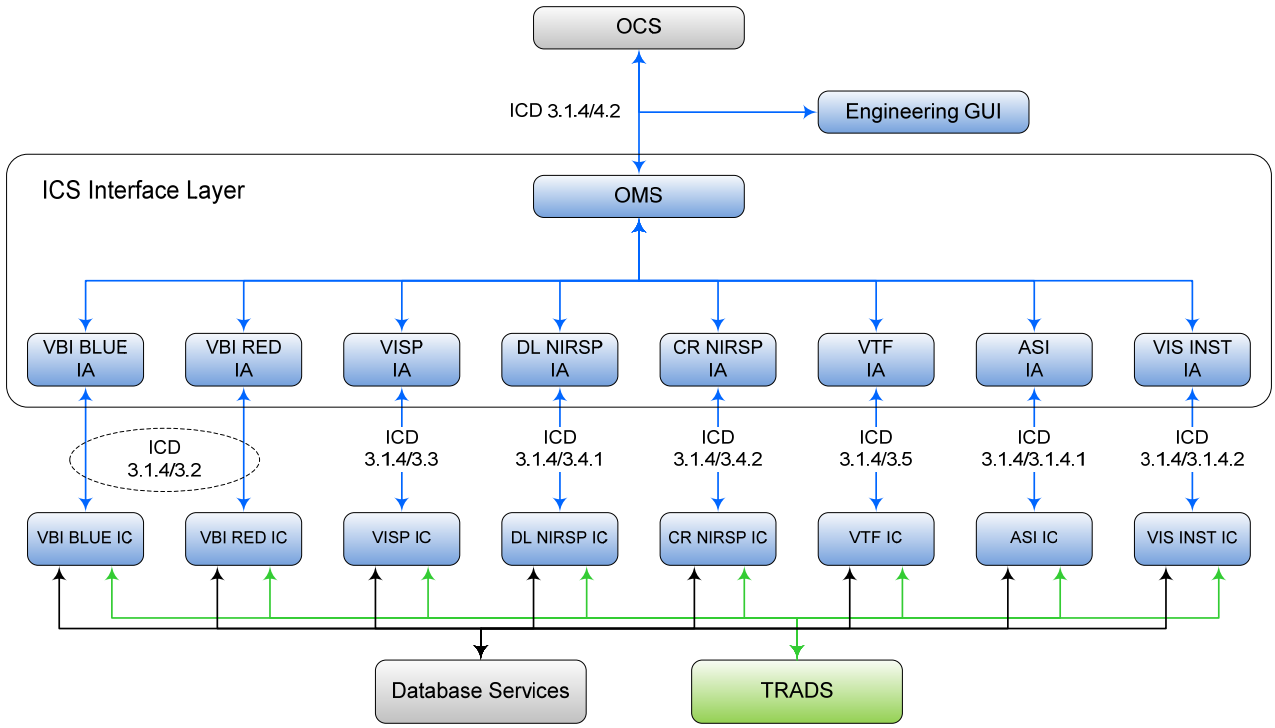


Figure 2: A block diagram of the Instrument Control System, illustrating the two main components, the Observation Management System (OMS) and the Instrument Adapters (IAs). Also shown are the Instrument Controllers (ICs) for the first generation ATST instruments, the Time Reference and Distribution System (TRADS), and the Database Services. Interfaces between the IAs and ICs define the specific observing parameters used by the instrument and passed down from the OCS.

appropriate observing parameters to the instruments, and typically include the following: a set of global experiment parameters, a list of instruments participating in the experiment, a list of those instruments that must complete their actions before the operation is considered complete, and all relevant parameters set by the user for each of the instruments. The ICS parses and interprets these configurations, extracting and forwarding the appropriate portions to each instrument in the observation. It checks on the readiness of the instruments and instructs each one to set itself up for the selected observing mode according to the parameters in the configuration. The ICS notifies the OCS when all instruments are properly setup (or not) for the observing mode.

As soon as the TCS is in position, the OCS notifies the ICS that the instruments may begin their individual observing actions for this mode (this may occur before the instruments have completed their setup activities). As soon as each instrument has completed its setup, the ICS notifies it to begin its observing actions. The ICS then monitors the instrument controllers for completion and sends status information back to the OCS.

After the OCS has notified the ICS that the instruments may begin observing, it may send one or more configurations to the ICS in preparation for the next Observing Block. The parameters in these configurations will be sent to the appropriate instruments as soon as possible, allowing them to setup for the next observing block as soon as they have completed the current one. This is referred to as *pre-configuration* and is done to maximize the efficiency of the observing process.

The ICS monitors the completion status of the participating instruments. When all of the instruments which are required to complete their observations for this observing block have completed their observing actions, the ICS terminates the observing activities of the remaining participating instruments and sends the final completion notification to the OCS.

New observing blocks are executed by repeating the pattern described above. This process repeats until all the observations for an experiment are completed or the experiment is cancelled or aborted by the operator.

INSTRUMENTS

The control of individual instruments is implemented by the ATST instrument developers. To facilitate the development of the instrument control software by these institutions, the ATST software team has developed a set of standard controllers and components that may easily be used to implement an Instrument Controller. This is called the Standard Instrument Framework (SIF). The SIF components are implemented using the Base and CSF components described earlier and consist of Management Controllers that control several other sub-controllers, Mechanism Controllers that control several Motion and Hardware Controllers, Detector Controllers that control one or more Camera systems, and an Instrument Sequencer that provides scripting support using an

embedded Jython scripting engine. Runtime access to observing scripts and parameter sets are provided through CSF database services. Motion Controllers and Hardware Controllers for control of standard motion control devices and hardware devices are provided as well. The Time Reference and Distribution System (TRADS) provides support for high resolution synchronization of instrument activity with absolute time and is the primary means of coordinating instrument mechanisms, cameras and polarization modulators with each other.

The ATST Instrument Partners are free to implement their Instrument Controllers using as desired using these tools as long as they meet the ATST Facility Instrument Interface.

Example of Instrument Control

Instruments are controlled by sending them CSF *configurations,* which are tables containing all the necessary attributes to execute a particular observing mode. As an example, the Visible Broadband Imager (VBI), which is a high-speed imager with a filter wheel, camera position and focus adjustment, might use the following attributes to execute a typical observation:

Table 1: Visible Broadband Imager Attributes

Name	Meaning
.experimentId	A unique ID for the experiment
.observationId	A unique ID for the observation
.obsMode	The current observing mode
.tscConfigured	A flag indicating if the telescope is in position and observing may begin
.paramSets	A list of parameter sets to be used
.paramSetSeq	The sequence in which parameter sets are to be used
.scriptName	The observing script to be run
.numCycles	The number of times to cycle through the parameter sets
.continueFlag	A flag indicating whether to continue after all cycles have been completed

A parameter set is a named collection of commonly used parameters that is downloaded from a database at run-time and passed to the script engine as an attribute. In this example a parameter set might contain the following parameters:

- The desired filter
- A map of camera field positions
- Camera setup parameters:
  - Exposure time
  - Number of frames
  - Binning
  - Region(s) of interest
- Data processing plug-ins to use

At run-time, a *configuration* with the above attributes is sent to the instrument. The instrument begins by setting all of its various devices into the proper position or mode as specified by the attributes. When everything is in position, the instrument checks to see if the telescope is in position and ready for observing to begin. If so, the observing script is run using the parameter sets and data recording commences. If not, the instrument waits patiently for the *.tcsConfigured* flag indicating it is time to start.

STATUS

The split in coordination roles, where the OCS handles high-level coordination between the TCS and ICS while the ICS handles the detailed coordination among a set of active instruments simplifies implementation of these tasks. An end-to-end simulation has been developed to test system behavior and a number of experiments have been run through this simulation, illustrating the functional correctness of this approach. The support provided by CSF at all levels of the control flow has greatly reduced the amount of specialized code that is required.

ACKNOWLEDGEMENTS

The National Science Foundation (NSF) through the National Solar Observatory (NSO) funds the ATST project. The NSO is operated under a cooperative agreement between the Association of Universities for Research in Astronomy, Inc. (AURA) and NSF.

The ATST represents a collaboration of 20 plus institutions, reflecting a broad segment of the solar physics community. The NSO is the Principal Investigator (PI) institution, and the co-PI institutions are the High-Altitude Observatory, New Jersey Institute of Technology's Center for Solar Research, University of Hawai'i's Institute for Astronomy, and the University of Chicago Department of Astronomy and Astrophysics.

REFERENCES

[1] See http://atst.nso.edu

[2] B. Cowan, S. Wampler, "Technologies for High Speed Data Handling in the ATST", *Astronomical Data Analysis Software and Systems XX (ADASSXX)*, Volume 442, (2011)

[3] S. Wampler. "A middleware-neutral common services software infrastructure." *10th ICALEPCS International Conference on Accelerator & Large Experimental Physics Control Systems*, Geneva, 10-14 October 2005.

[4] B. Goodrich, S. Wampler, "Execution of configurations using the ATST controller", *Advanced Software and control for Astronomy*, Editors: H. Lewis, A. Bridger. Proc. SPIE 6274, 62741X (2006).

[5] J. Hubbard, B. Goodrich, S. Wampler, "The ATST Base: Command-Action-Response in Action", in *Software and Cyberinfrastructure for Astronomy*. Editors: N.M. Radzwill, A. Bridger. Proc. SPIE7740, 77402R (2010).

THE LABVIEW RADE FRAMEWORK DISTRIBUTED ARCHITECTURE

O. Ø. Andreassen, D. Kudryavtsev, A. Raimondo, A. Rijllart, V. Shaipov, R. Sorokoletov, CERN, Geneva, Switzerland

Abstract

For accelerator GUI applications there is a need for a rapid development environment to create expert tools or to prototype operator applications. Typically a variety of tools are being used, such as Matlab or Excel, but their scope is limited, either because of their low flexibility or limited integration into the accelerator infrastructure. In addition, having several tools obliges users to deal with different programming techniques and data structures.

We have addressed these limitations by using LabVIEW, extending it with interfaces to C++ and Java. In this way it fulfils requirements of ease of use, flexibility and connectivity, which makes up what we refer to as the RADE framework.

Recent application requirements could only be met by implementing a distributed architecture with multiple servers running multiple services. This brought us the additional advantage to implement redundant services, to increase the availability and to make transparent updates.

We will present two applications requiring high availability. We also report on issues encountered with such a distributed architecture and how we have addressed them.

The latest extension of the framework is to industrial equipment, with program templates and drivers for PLCs (Siemens and Schneider) and PXI with LabVIEW-Real Time.

INTRODUCTION

Integrating equipment and software in CERN accelerators like the LHC, which contains more than 1200 dipole magnets, more than 8000 other superconducting magnets operating at temperatures down to 1.9 K and with currents up to 12000 A is a great challenge [1]. When one also adds its size into the equation (27 Km) it doesn't ease the task of operating this marvellous machine.

With the RADE (Rapid Application Development Environment) framework and LabVIEW we have managed to ease the job of integrating new and existing projects into all the different disciplines at CERN [2]. With our distributed architecture the user does not have to update their RADE libraries, we maintain them on our server and ensure compatibility with new releases as long as the underlying protocol or library does not become deprecated or replaced.

ARCHITECTURE

In computer programming, a software framework is an abstraction in which software providing generic functionality can be selectively changed by user code, thus providing application specific software. It is a collection of software libraries providing a defined application-programming interface (API) [3].

LabVIEW already gives you the foundation needed to do fast programming by providing an integrated development, debug and deployment environment based entirely on graphical programming, as well for the code, as for the GUI development. In addition it gives you a powerful and flexible set of analysis and math libraries out of the box.

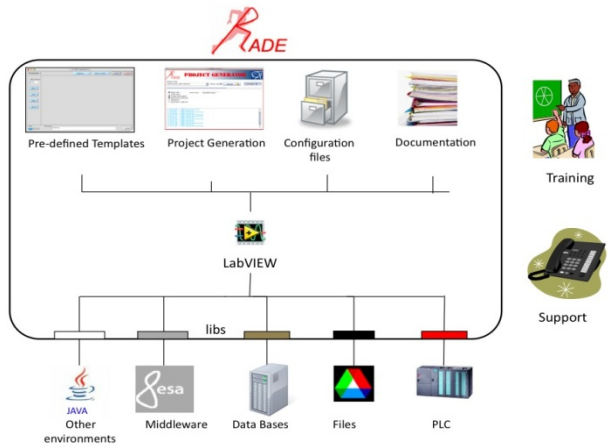


Figure 1: RADE Architecture.

Our development effort is thus concentrated on the integration into the CERN accelerator control and storage infrastructures, with an objective of making the deployment and maintenance of the framework transparent for the users, yet always up-to-date with the latest dependencies and additions.

Total Package

The RADE framework also aims to give users a total package for development, maintenance and support (Fig 1), making it quick to implement yet highly stable, maintainable and flexible through well defined development templates, guidelines and documentation.

Currently we are working on expanding this framework and do a "total integration" in LabVIEW at CERN.

Templates, documentation, source control, updates, and libraries are being added to the foundation as the testing of them finishes.

By adding a layer on top of LabVIEW we can synchronize the client installation with our Subversion (SVN) [4] and data repositories.

Project Generator

In addition a project generator (Fig. 2) has been created which can be called through the native menus of LabVIEW.

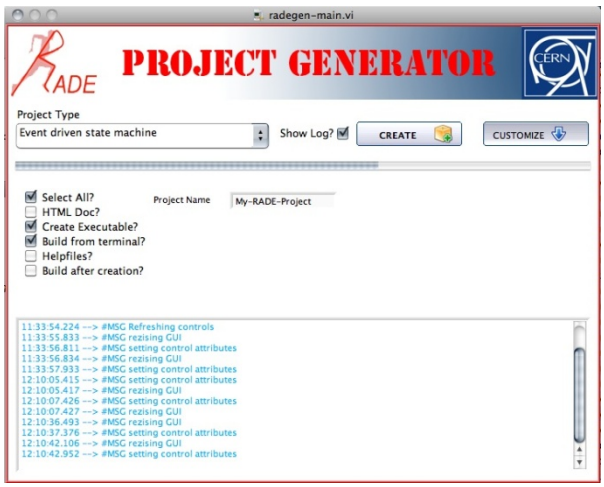


Figure 2: RADE Project Generator.

The Project Generator will create a skeleton project and folder structure for the user, and if desired store this project on our SVN repository dedicated for user projects. It also automatically generates documentation, executable files, help files, and integrates system and RADE components into the project so that if the user at a later point would like to update or retrofit his sources to a later version of RADE, he can do so.

RADE LIBRARIES

The RADE libraries consist of several communication layers (Fig. 3), adapted to the necessary protocols. It also makes use of a distributed architecture where several redundant servers host the communication and analysis libraries. These can potentially be written in any software language, but we mainly use the CERN Java packages and native LabVIEW servers that communicate with the LabVIEW client through sockets.

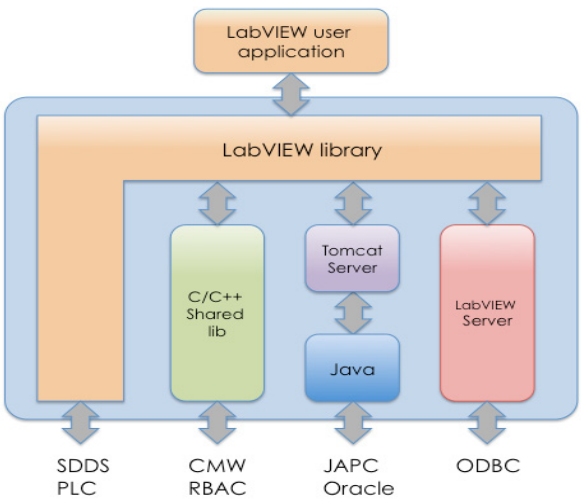


Figure 3: Library layers in RADE.

Java interface

The Java API for Parameter Control (JAPC) is a communication layer to control accelerator devices from JAVA [5]. Client programs can access JAPC parameters with set and get actions or by subscribing to the data. JAPC is a unified Java API for almost all parameters present in the control system. The diversity of devices is handled below the JAPC interface where each kind of device has its own implementation.

In the RADE framework we have implemented a LabVIEW to JAPC interface using a Tomcat server (Fig. 3) as mediator. The same mechanism is used to access ORACLE databases.

CMW Wrapper

The Common MiddleWare (CMW) API incorporates Set, Get and monitor actions like the JAPC API, but it is written in C++ and runs locally with the client [6].

We implemented the LabVIEW to CMW interface (CMW wrapper) by wrapping the C++ API into call back functions and turning it into a shared library, which is called by LabVIEW using the call library interface node.

The Toolkit was designed to be cross platform so the dependant libraries can be compiled on any platform supported by the CMW API (Mainly Linux and Windows).

SDDS Library

SDDS stands for Self Described Data Set [7]. This file format is commonly used at CERN and can contain any kind of data. This standard is used in the LHC to store equipment data (currents, voltages, levels etc.), used to inform the domain experts about the system health.

Since a failure in the machine can produce more than 10.000 files in a single dump, the SDDS library was designed to index and read only the requested portion of a file, improving reading speed and saving time when performing analysis.

PLC Library

The communication to the PLC is written in standard LabVIEW using the "Fetch-Write" protocol from Siemens through TCP-IP. The "Fetch-Write" has to be declared in the PLC to authorise external devices accessing its data blocs. For more flexibility we are developing a wrapper for LIBNODAVE [8], an open source library to access Siemens PLCs.

RADE IN THE LHC DASHBOARD

The LHC Dashboard project aims to provide a single entry point to the monitoring data collected from the experiments and equipment associated with the LHC.

The Dashboard collects information from multiple sources, treats the data and renders it as a set of web

pages. It covers various activities of the LHC making up a complete picture of what goes on in the machine. By using RADE in the dashboard we have managed to make a very flexible and powerful data-mining tool where we, with little effort, can obtain new data and display this for our users on demand. Currently development is ongoing using RADE for expanding the dashboard, making a fully flexible, user driven web interface where we can render any data from the various activities at CERN on request. The page will consist of a large library of gadgets where the users can select what to see in a customized view.

RADE IN HARDWARE COMMISSIONING

Hardware Commissioning is the phase where tests are carried-out by the specialized teams to qualify the individual systems of the LHC for operation (vacuum, cryogenics, quench protection, interlocks, powering, etc.). The interaction of each system and its partners are verified and the circuit will be powered up to nominal current [9]. This is repeated for every circuit. Commissioning the LHC requires powerful diagnostics to identify faults in the equipment protection systems. This diagnostics tool, called herein the post-mortem system (Fig. 5), has the role of grouping and analysing transient data recorded in the LHC equipment.

The Hardware Commissioning software architecture consists of three parts. The first is the Sequencer written in JAVA, executing the tests. The second is the data collection and storage. The third is the data analysis made with RADE and LabVIEW.

With RADE the Post Mortem Framework is highly flexible. It allows us to quickly adapt to changes and new demands, and validate them on a set of reference tests, when performing the crucial analysis of the LHC equipment. All RADE facilities have been used during the implementation of the Post Mortem analysis.

In 2011, 6092 tests where executed (Fig. 6) (3 week campaign). From these, 3204 where automatically approved by software using RADE.

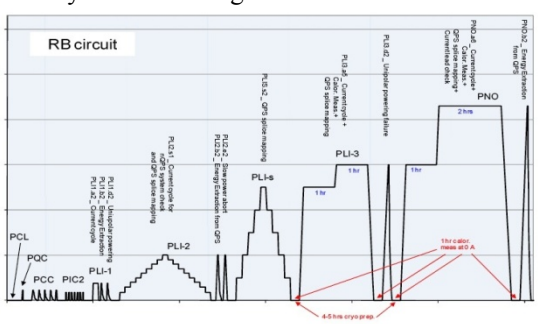


Figure 6: HWC Dipole Tests.

ISSUES

Having a distributed architecture is highly flexible. You can re-route clients to one server while performing updates on the other. On the other side one is vulnerable as well. If the server goes down, if there is a critical bug

Figure 4: The LHC Dashboard.

in the software running or by any other means a problem in the chain that blocks, all users are affected.

Since most of the software dependencies are developed and maintained by third parties, who again have dependencies on other repositories and machines, we have to make sure that the source running on the server is up to date and in synch with versions. Since the servers are redundant, users will transparently connect to the available source. This can be a problem since a

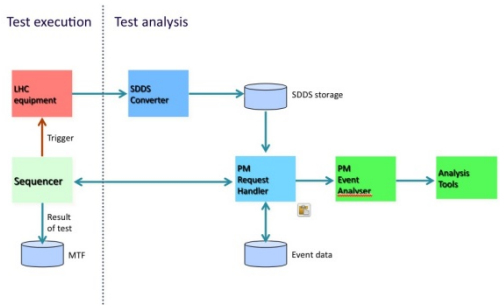


Figure 5: Post Mortem System.

connection or request that manages to block the server could potentially be re-directed to the secondary machine and block that one as well if executing the same request.

To cope with these issues we have to constantly analyze the system integrity and add logic, which identifies problems and prevents them from happening.

CONCLUSIONS

The flexibility and short development time experienced when using RADE in, amongst others, the presented project examples show that the RADE framework is an excellent and powerful tool that can be used to cope with challenges in an environment that quickly and constantly changes. One has however to take care when using a distributed architecture since introduction of new "features" or changes can quickly cause unforeseen problems that affect many users. On the other side the flexibility and convenience of having a distributed architecture trumps the downsides by far.

REFERENCES

[1] L. Evans, "The Large Hadron Collider – Present Status and Prospects", IEEE Trans. Appl. Supercond., Vol. 10 No. 1 (2000), 44-48
[2] A. Raimondo, "Rapid Application Development Environment Based on LabVIEW" - edms.cern.ch/document/904425/1
[3] Definition software framework http://en.wikipedia.org
[4] Subversion - subversion.apache.org
[5] V. Baggiolini, "JAPC-the java API for parameter control", ICALEPCS2005, Geneva, Switzerland.
[6] K. Kostro, "The control middleware (CMW) at CERN status and usage", ICALEPCS2003, Gyeongju, Korea.
[7] R. Soliday, "New features in the SDDS tool kit", PAC2003, Portland, Oregon, US.
[8] LIBNODAVE, "Exchange data with Siemens PLCs" http://libnodave.sourceforge.net
[9] R. Saban, "LHC Hardware Commissioning Summary", EPAC08, Genoa Italy.

INTEGRATING ETHERCAT BASED IO INTO EPICS AT DIAMOND

R. Mercado, I. Gillingham, J. Rowland, K. Wilkinson
Diamond Light Source, Oxfordshire, UK

Abstract

Diamond Light Source is actively investigating the use of EtherCAT-based remote I/O modules for the next phase of photon beamline construction. Ethernet-based I/O in general is attractive, because of reduced equipment footprint, flexible configuration and reduced cabling. EtherCAT offers, in addition, the possibility of using inexpensive Ethernet hardware, off-the-shelf components with a throughput comparable to that of current VME-based solutions. This paper presents the work to integrate EtherCAT-based I/O to the EPICS control system, listing platform decisions, requirements considerations and software design, and discussing the use of real time pre-emptive Linux extensions to support high-rate devices that require deterministic sampling.

INTRODUCTION

Diamond Light Source is a third-generation synchrotron light source which started operations in 2007. The current operational state includes twenty photon beamlines, with a further twelve beamlines due to be completed by 2017. Of these, three are in advanced stages of design and construction.

Diamond's control system is based on the EPICS control system toolkit[1][2]. In the current control system, generic I/O is interfaced through a range of VME hardware. This architecture is being reconsidered in the context of the next phase of beamline construction [3]. In the new architecture most I/O functionality, including motion control, video and I/O for analogue and digital signals, will be realised through Ethernet-attached I/O.

Ethernet-based I/O allows the replacement of VME crates by 1U x86 PC as the EPICS input output controller (IOC) servers. This saves rack space and uses easily replaceable standard computing server hardware, available from many manufacturers; configuration is made more flexible because signals do not need to be concentrated in a single crate. The use of Cat6 cabling is an additional advantage.

ETHERCAT

EtherCAT[4][5] is a real-time Ethernet protocol that relies on conventional Ethernet frames with very short cycle times and efficient bandwidth utilisation. The protocol uses the full duplex mode of Ethernet; each communication direction is operated independently of the other.

EtherCAT operates with a master that passes EtherCAT telegrams to a series of EtherCAT slaves. The EtherCAT master uses standard Ethernet controller hardware, whilst the slaves use a custom slave controller. The slaves process the incoming telegrams directly; they extract or insert user data and transfer the telegrams to slaves downstream, each slave introducing a delay of a few nanoseconds. The last EtherCAT slave automatically returns the processed telegram back to the master as a response telegram.

Each EtherCAT slave includes a controller with a Fieldbus Memory Management Unit (FMMU), which allows the mapping of logical addresses in the telegram to physical ones within the slave. The FMMU converts logical addresses to physical ones via an internal table, configured at slave initialisation. It is able to address physical locations down to the level of individual bits and is configured at start-up.

The telegram structure, combined with the FMMU capabilities, allows several slaves to be addressed in a single Ethernet frame. This characteristic significantly reduces the overhead in comparison to other Ethernet fieldbus protocols and is well suited to addressing devices that may have a payload of only a few bytes, such as digital I/O devices, typical of industrial automation.

The registers in each slave that can be mapped by the FMMUs are known as Process Data Objects (PDOs). After configuration the primary EtherCAT telegram present on the bus is Logical Memory Read and Write (LRW). This exchanges PDO data bi-directionally between the master and multiple slaves.

EPICS SOFTWARE COMPONENTS

Diamond EPICS soft IOCs typically run on x86 Linux servers. Prior to interfacing EtherCAT, there had not been a requirement for deterministic sampling on this platform, as operations that relied on more precise timing were delegated to dedicated VME hardware. x86 servers therefore typically ran a standard RedHat Linux kernel. To enable the deterministic performance of EtherCAT, a Real Time Linux kernel was required for servers running the EtherCAT master. The details are described in the next section.

A further requirement that was considered was the need to segregate functionality into separate IOCs that could be maintained separately, for example for separate technical areas. To realise this, multiple EPICS IOC instances needed to have access to every EtherCAT bus scanner. The resulting architecture is shown in Figure 1.

In the physical realisation, a server has a minimum of two Ethernet interfaces, one for standard TCP/IP traffic and an EtherCAT-dedicated interface. There is a single master for every EtherCAT-dedicated interface. The current deployments have one such interface per server, but more could be used if the application requires further segregation of components within the same EPICS IOC server.

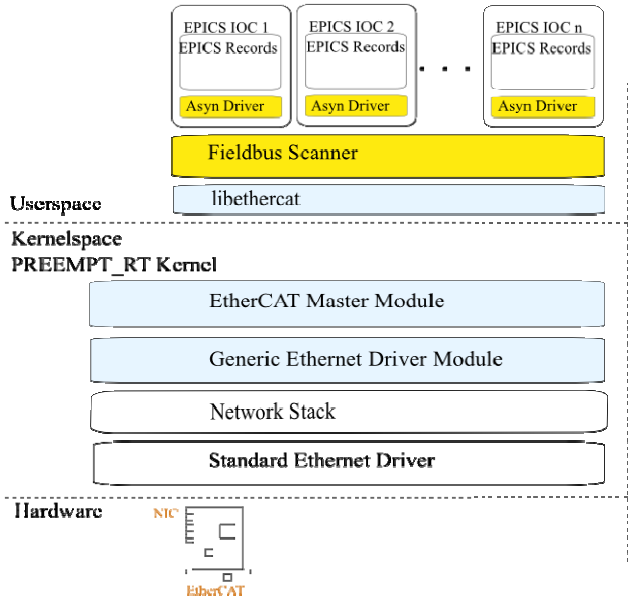


Figure 1: Software components. Cyan represents elements from Etherlab, whilst yellow represents elements from Diamond.

REAL-TIME LINUX

Timing jitter in the EtherCAT bus master process results in irregular bus cycles, which result in missed samples. A standard Linux kernel running a desktop configuration on a multi-core machine misses approximately 0.1% of bus cycles when running at a bus rate of 1 kHz, as measured using a slave device with an internal clock and cycle counter.

The kernel patch pre-empt real-time (PREEMPT_RT) reduces latency and adds the ability to pre-empt most kernel critical sections. It is actively developed by RedHat, and commercial support is available as the MRG product [6]. PREEMPT_RT was chosen over harder real-time Linux systems such as RTAI [7] or Xenomai[8] because of the standard API, minimally disruptive installation and acceptable performance for EtherCAT applications.

Installation of PREEMPT_RT is realised by means of a kernel RPM and configuration scripts to assign kernel thread priorities. The user space APIs are unchanged as the necessary calls are already present as part of the POSIX ADVANCED REALTIME standard[9], so that applications can be developed and tested on a standard system, albeit with reduced performance. For this application Diamond is using kernel 2.6.33.9-rt31, from MRG 1.3.

The following API features are used in the EtherCAT master:

- high-precision timers using clock_nanosleep
- mlockall, to prevent the process image paging
- SCHED_FIFO pthread scheduling
- PTHREAD_PRIO_INHERIT mutexes

No missed cycles are observed when using PREEMPT_RT, under CPU loaded conditions and an EtherCAT bus rate of 1 kHz. Table 1 shows the difference in latency measured using the cyclictest tool. Note that Kernel 2.6.18 is the standard kernel for RHEL5 and does not have high-precision timers, kernel 2.6.33 has high-precision timers but is not fully pre-emptable, and kernel 2.6.33-rt31 has the PREEMPT_RT patch applied.

Table 1: Timing Latency

Kernel	Mean	Max
2.6.18	1630 us	2745 us
2.6.33	52 us	243 us
2.6.33-rt31	5 us	54 us

ETHERCAT MASTER

Diamond uses the EtherCAT master from etherlab.org[10], an open-source kernel module that implements the EtherCAT master state machine and FMMU configuration. The generation of bus cycles is left to the user. This allows the master to work with a variety of different real-time Linux implementations. The master also provides patched network drivers for some common network cards to support interrupt-free operation and thus reduced latency. However these were not required as the generic interface, which uses PF_PACKET raw sockets, provides acceptable performance. To achieve this, it is necessary to disable interrupt coalescing in the network driver to allow high bus rates (up to 10 kHz).

The EtherCAT master modules have been packaged into a Dynamic Kernel Module Support (DKMS) RPM that is built automatically at boot time if the kernel is upgraded.

BUS SCANNER

The Bus Scanner generates bus cycle packets at a steady rate and multiplexes EtherCAT bus access between clients over a pre-threaded UNIX domain stream socket for inter-process communication (IPC). This provides functional isolation at the record level by allowing multiple EPICS IOCs to share the same bus, but restart separately. Broadcast protocols such as UDP or shared memory were not chosen because of the low number of expected clients and the added complexity of implementing connection management and reliability. Real-time and non-real-time tasks are decoupled by the use of message queues implemented using pthread mutexes and condition variables, with priority inheritance to prevent priority inversion.

On start-up the Bus Scanner reads a configuration file, configures the FMMUs, and puts the slave state machines to into operational (OP) mode. The bus cycle timer is then started, and every 1ms the LRW frame is sent and received. The PDO data is pushed on to each client queue in non-blocking circular buffer mode. Write commands

from each client are added to the Scanner command queue in blocking mode, and merged into the local copy of PDO memory ready for the next bus cycle. The Bus Scanner also distributes the FMMU assignments to the clients on connection, as the clients are responsible for unpacking the PDO memory.

Scanner Configuration

The scanner configuration XML file is generated from a bus configuration XML file and a directory of EtherCAT Slave Information (ESI) files provided by the slave vendors. The bus configuration maps bus positions to unique names and also selects the oversampling rate for the Beckhoff eXtreme Fast Control (XFC)[10] devices that can produce more than one sample per bus cycle, as below:

```
<chain>
  <device type_name="EL2004" revision="0x00100000"
position="1" name="OUT0" />
  <device type_name="EL3702" revision="0x00020000"
position="2" name="RF0" oversample="11" />
</chain>
```

The ESI files describe the named registers present in each slave.

ASYN DRIVER

The EPICS device support for the EtherCAT Scanner uses the asynPortDriver C++ class. On iocInit the bus configuration is read over the UNIX socket used for IPC. One port is created for each slave, and one for the bus master status. Port names for each device and asynInt32 parameters for each register are automatically generated from the device names assigned in the bus configuration file and the register names in the ESI file. Additional ports can be instantiated to support oversampling devices and to add triggered circular buffers to any channel. Test templates are generated from the ESI file for each device using the Python libxml2 library, but the final template for each device is hand-written to comply with record naming and interface conventions.

EtherCAT port drivers implement the ProcessDataObserver interface. The socket read thread maintains a list of observers and calls a method on this interface to deliver PDO data on each bus cycle. The port driver unpacks the appropriate registers from the PDO data using the PDO mapping information provided by the scanner on connection, and writes the asynPortDriver parameter cache. An epicsMessageQueue delivers write commands from the port drivers to the socket write thread.

The master port driver reports network cable link status, number of connected slaves and slave state mode bits. One limitation of the current device support is the inability of the asynPortDriver to return an alarm; this will be fixed in a future release of Asyn.

PROGRESS TO DATE

The EPICS implementation was tested with slaves from Beckhoff (Verl, Germany), SMC Pneumatics (Tokyo, Japan) and National Instruments (Austin, TX, USA). Most slaves tested are from Beckhoff, and these come with comprehensive ESI files. The National Instruments backplane needed its configuration EEPROM to be pre-programmed before it could be used with the Etherlab master. Products from SMC Pneumatics have limited availability of slave information files, and for National Instruments, the re-configurable EtherCAT backplane does not lend itself to a fixed file entry. Diamond is also investigating with National Instruments whether higher data rates can be supported as with Beckhoff XFC slaves.

As part of Diamond's latest build phase, EtherCAT technology is being incorporated into new beamline front-ends. The build tree for each IOC incorporates generation of the scanner configuration XML file. A Python script is used to expand the simplistic XML description in the build tree into the full XML configuration file. All the Asyn ports are auto-generated at start-up, simplifying the IOC boot script.

The transition from VME based IOCs, running VxWorks to Linux PC IOCs with EtherCAT, has been relatively straightforward. Very few changes were required to the client EPICS applications. Modifications mainly consisted of substituting the new Asyn port names in place of VME I/O references, along with the creation of new, simpler IOC boot scripts.

CONCLUSION

EtherCAT was integrated with the EPICS control system toolkit on a Linux Real-time platform. EtherCAT devices are configured and scanned using the Etherlab open-source master. The bus scanner communicates with an Asyn driver making I/O signals and configuration information available to EPICS IOCs. The bus scanner broadcasts the PDOs to several soft IOCs that can run in the same server for segregation of technical areas. The configuration in the scanner process is reused in the Asyn driver. The driver automatically creates at start-up one port per slave and one port for the master status.

The solution is being adopted for future control system deployments in the next phase of photon beamline construction at Diamond Light Source.

REFERENCES

[1] M.T. Heron et al. “The Diamond Light Source Control System”, EPAC 2006, Edinburgh, June 2006, THPCH113, p. 3068 (2006); http://www.JACoW.org.

[2] M.T. Heron et al. “Implementation, commissioning and current status of the Diamond Light Source Control System”, ICALEPS 2007, Knoxville, Tennessee, USA, October 2007, TOAA05, p. 56 (2007); http://www.JACoW.org.

[3] I.J. Gillingham et. al, "Diamond's Transition from VME to Fieldbus Based Distributed Control", PCaPAC 2010, Saskatoon, Canada, October 2010, THCOAA04, p. 124 (2010); http://www.JACoW.org.
[4] International Electrotechnical Commission, "Industrial Communication Networks Fieldbus specifications Part 3-12: Data-link layer service definition – Part 4-12: Data-link layer protocol specification – Type 12 elements", IEC, 61158-3/4-12:2007
[5] D. Jansen and H Büttner, "Real Time Ethernet: the EtherCAT Solution", Computing and Control Engineering Journal, 2004 **15(1),** p. 16
[6] Red Hat Enterprise MRG Documentation. http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_MRG [Accessed Aug 2011]
[7] Dipartimento di Ingegneria Aerospaziale - Politecnico di Milano "Real Time Application Interface - RTAI" https://www.rtai.org/ [Accessed Sep 2011]
[8] Xenomai: Real-Time Framework for Linux http://www.xenomai.org [Accessed Sep 2011]
[9] IEEE and The Open Group "The Open Group Base Specification Issue 6" http://pubs.opengroup.org/onlinepubs/009695399 [Accessed Sep 2011]
[10] IgH EtherCAT master for Linux. http://www.etherlab.org [Accessed Aug 2011]
[11] Beckhoff "eXtreme Fast Control Technology." http://www.beckhoff.com/english/default.htm?highlights/xfc/components.htm [Accessed Sep 2011]

THE ATLAS TRANSITION RADIATION TRACKER (TRT) DETECTOR CONTROL SYSTEM

Jolanta Olszowska, Elżbieta Banaś, Zbigniew Hajduk, IFJ-PAN, Cracow, Poland
Tadeusz Kowalski, Bartosz Mindur, AGH University of Science and Technology, Cracow, Poland
Ruslan Mashinistov, Konstantin Zhukov, LPI, Moscow, Russia
Anatoli Romaniouk, MEPhI, Moscow, Russia
Michael Hance, Dominick Olivito, Peter Wagner, University of Pennsylvania, Philadelphia, Pennsylvania, U.S.A.

Abstract

The TRT is one of the ATLAS experiment Inner Detector components providing precise tracking and electrons identification. It consists of ~350000 proportional counters (straws) which have to be filled with stable active gas mixture and high voltage biased. High voltage settings at distinct topological regions are periodically modified by closed-loop regulation mechanism to ensure a constant gaseous gain independent of drifts of atmospheric pressure, local detector temperatures and gas mixture composition. Low voltage system powers front-end electronics. Special algorithms provide fine tuning procedures for detector-wide discrimination threshold equalization to guarantee uniform noise figure for whole detector. Detector, cooling system and electronics temperatures are continuously monitored by ~ 3000 temperature sensors. The standard industrial and custom developed server applications and protocols are used for devices integration into unique system. All parameters originating in TRT devices and external infrastructure systems (important for Detector operation or safety) are monitored and used by alert and interlock mechanisms. System runs on 11 computers as PVSS (industrial SCADA) projects and is fully integrated with ATLAS DCS.

INTRODUCTION

TRT Detector

The TRT (Transition Radiation Tracker) is a part of the Inner Detector [1] tracking system of ATLAS experiment built at CERN for the Large Hadron Collider (LHC). It covers $|\eta|<2.5$ in pseudo rapidity at radii between 128 and 206 cm, allows for electron-pion separation at 97% level and provides continuous tracking with accuracy ~120 μm/point. The detector has three parts – barrel and two end-caps – which consist of arrays of the thin walled proportional counters – straw tubes. The barrel detector contains 96 parts called modules (3 layers of 32 modules). The end-caps consist of 14 'wheels' each, where wheel is functionally divided into 32 sectors. Each module and sector has its own individual electrical services (HV, LV, timing etc). The straw configuration gives ~36 space measurement points in whole TRT acceptance. The detecting elements act as drift tubes ensuring the tracking function and measure energy deposit in specially chosen active gas allowing for efficient detection of the transition radiation photons thus performing identification function.

ATLAS Detector Control System (DCS)

The ATLAS DCS is a large distributed hierarchical control system integrating all subdetectors and infrastructure systems. PVSSII from ETM [2] is a commercial product chosen as a standard SCADA for CERN control systems. The CERN Joint Controls Project (JCOP) Framework [3] provides guidelines and components for integration of commonly used hardware and services into a homogenous control system. ATLAS Central DCS group provides ATLAS-specific guidelines and framework components. The Finite State Machine (FSM) [4] component is used to build and integrate ATLAS DCS hierarchy providing scalability and powerful tools for partitioning of large systems. The main FSM building blocks modelling behaviour of hardware objects are Device Units (DU) and groups of devices are Control Units (CU). States and statuses propagate up in the FSM hierarchy, while commands propagate down the tree. For each FSM node in a hierarchy, two GUI operator panels (primary and secondary) have to be developed. Operator panels can be extended by expert oriented elements controlled by Access Control. The effort needed for design and construction of the control system for LHC experiments in many cases surpasses the possibilities and available manpower of groups involved. This has resulted in wide use of commercial equipment and software tools making the tasks possible to achieve. OPC (OLE for Process Control) standard has been adopted by CERN as one of the mandatory specifications for commercial equipment purchased for LHC experiments.

THE TRT DCS

The TRT DCS subsystems controlling hardware directly connected to the detector are organized to follow Data Acquisition architecture – End Cap A , End Cap C, Barrel A and Barrel C FSM nodes. Each of them contains High Voltage, Low Voltage and Temperature nodes. All other FSM sub-trees are grouped under an Infrastructure node. Four PCs control LV Distribution and Temperatures Monitoring Systems, three PCs control HV

Distribution System, one PC is assigned to infrastructure and whole hierarchy is supervised by top located PC.

High Voltage Distribution System

The TRT High Voltage is based on HVSys company Multichannel Crates with SC508 controllers [7]. The SC508 controller provides communication through RS-232 interface with custom designed protocol. There is no parameters' polling loop embedded in HVsys crate firmware. A boot up message is the only one which the crate sends out autonomously. Six HVSys crates are located in ATLAS underground counting room, 1984 individual lines of length ~70 m distributes HV for both End Caps and Barrel. Three PVSS projects control ~14000 set-point registers and monitor ~18000 measured values and statuses.

A custom OPC DA server was developed (using Softing OPC Toolbox) as an integration middleware for TRT HV PVSS projects. SC508_OPC supports calibration files for Digital-to-Analogue Converter (DAC) and Analogue-to-Digital Converter (ADC) of each HV Cell. One of the main difficulties during commissioning was tuning of performance of HV PVSS systems. The 2-port PCI/RS232 interface cards showed to be more efficient communication solution than USB HVSys crates interfacing. The separate OPC server threads are handling each of serial ports. Separate "fast transaction" queues support commands and status information handling. Each OPC server handles 9 OPC groups defined in order to spread in time PVSS Event Manager load during periods of full TRT Detector transition between HV operational states.

The FSM for HV systems adopts majority logics for states decoding. It allows a small fraction of cells to be tripped or recovering without data taking disturbances.

Two guarding mechanisms were added when an operational experience has been cumulated:

- There are values stored locally in HV cells which do not need to be changed during states transitions. They define cell's behaviour during over-current hardware trip procedure. Trip registers values are refreshed to prevent their corruption - one cell every 3 min.
- A watchdog covering a full communication path for each of HV crate was developed. One HV spare cell per crate is used for it. Each 3 minutes a new HV set-point value is sent to a Watchdog Cell. The read back from set-point register and the output voltage value are compared with a previous watchdog setting. HV watchdog is a very sensitive tool for detecting cases when system suffers from PVSS OPC Client memory leaks, internal HV crate communication problems, OPC server operation slow down or other HV communication degradation.

Low Voltage Distribution System

The Low Voltage TRT system has been detailed elsewhere [8]. The TRT performance and good quality data readout depend on proper level of supply voltages, assuring the correct operation of the Front End (FE) electronics during life cycle of the experiment.

The data/control flow in the LV FE part during standard detector operation is the following:

- Output voltages, currents and intermediate board's (LVPP - located within ATLAS detector) temperatures are continuously monitored in the PVSS projects. These data are read out over CANbus via CanOpen OPC server from the ELMB [5] embedded on the board.
- Settings of DACs (defining output voltage level) are read back from the LVPP employing TRT custom Dynamic Link Library (DLL), embedded as an extension to PVSS CTRL scripting language. This readout is performed after every change of DAC values, before switching on voltages, and on demand.
- OCM (Over Current Monitor) values are read back from the LVPP with mentioned TRT custom DLL every 20 minutes.
- Values of supply voltages and temperature directly at the FE: Voltage/Temperature (VT) are read by the Data Acquisition (DAQ) system and are sent to the DCS via the DCS-DAQ Communication (DDC) mechanism. This readout is done at every run start and on demand.
- All operations (switching on/off, writing new DAC setting to the hardware, etc.) in the LV FE system are executed by FSM commands, available from PVSS FSM GUI. These commands are transmitted via CANbus employing already mentioned TRT custom DLL.

Temperatures Monitoring

The TRT detector operates in ambient temperature. It should be kept in proper temperature ranges because of detector and electronics safety and mechanical structure stability. The detector is exposed to heating coming mainly from FE electronics. It can also suffer from cooling which comes from the silicon detectors placed inside the TRT, when the insulating envelope heaters fail.

Two types of temperature sensors are installed in the TRT detector:

- NTC, mounted directly on the FE electronic boards.
- PT1000, mounted on different structures in the detector.

The following are groups of sensors installed on different parts of the detector:

- 1720 NTC's on FE boards - acting an important role in LV interlock
- 496 PT1000's on FE boards cooling pipes
- 160 PT1000's on mechanical structure of the detector
- 288 PT1000's between detector modules in Barrel and in straw's cooling CO_2 flow – acting as sensors for mean temperature calculations for GGSS to HV feedback.

- 48 PT1000's on heat exchangers cooling pipes in End Caps
- 32 PT1000's on input pipes of straws' cooling CO_2 flow in Barrel

Sensors' values are monitored by ELMB embedded on the TTC PP boards. Calibration constants were measured for each cable with a PT1000 sensor and are used in CanOpen OPC server configuration files.

Alarm limits are set individually for each sensor. They trigger emergency and notification actions and are used in temperature FSM nodes.

TRT Infrastructure Subsystems

There are following auxiliary subsystems fully integrated with TRT DCS, but not connected directly with a detector:

- Maraton Bulk Power Supplies which deliver primary LV for LVPP distribution boards (and other Patch Panel boards) and DAQ VME Crates are commercial devices delivered by WIENER company. A common OPCserver handles both types of devices, merging Maratons' Ethernet communication with CANbus one for VME crates. Framework components are supported by CERN EN/ICE group.
- The Hardware Interlock box for LV system is supervised by built-in ELMB's, thus it can be monitored using a standard middleware layer.
- CANbuses Power Supply Units deliver voltages for 25 CANbuses used in the TRT. Standard tools delivered by ATLAS DCS group were used for integration to TRT DCS. The above Infrastructure nodes are controlled from one PVSS system.
- To achieve the necessary performance of the TRT, the gas gain should be kept constant. It depends on actual active gas composition, concentration of impurities, ambient pressure, local detector temperature and high voltage bias which can be used as a parameter compensating drifts of other values. The dedicated standalone Gas Gain Stabilisation System (GGSS) was developed. The standalone process running in closed loop collects Fe^{55} spectra from reference straws with TRT active gas flowing through and corrects HV biasing sensors as to keep spectra peak positions at required level after every iteration. GGSS is a source of reference HV value and reference temperature which are used by TRT High Voltage Systems for periodical corrections of HV cell's output voltages. A detailed description of hardware architecture and a stabilisation algorithm can be found in [9]. A dedicated Distributed Information Management (DIM) server [6] integrated with the stabilization process is used to send stabilization results and status messages to PVSS system and receive commands from FSM. The normalized HV reference value calculation was added as a result of operational experience. It is a difference between measured reference HV and a predicted value calculated on base of atmospheric pressure and local temperature measured values. Normalized HV reference is a very sensitive parameter guarding active gas composition.
- The Active Gas Chromatograph is a commercial standalone system running on one PC and integrated with TRT DCS by periodically fetching analysis results from an output file.

ATLAS Infrastructure Subsystems

The TRT detector uses centrally delivered and maintained infrastructure hardware and services. It's very important to monitor parameters which directly can influence the detector operation. One can copy or subscribe needed data from Information Services Systems or PLC's which control external systems. The following external systems parameters sets are integrated within TRT DCS Infrastructure node:

- Active Gas System – delivers a gas mixture (70% Xe, 27% CO_2, and 3% O_2) circulating in closed loop system.
- Straws Cooling – removes heat dissipated in straws by ionisation. It circulates (CO_2) between straws.
- FE electronics & Cables Cooling – liquid (C_4F_{16}) circulation.
- Detector Safety System – displays triggers & action bits from hardware matrix.
- Common Infrastructure Control – controls racks and environmental sensors.
- PVSS Systems Overview – monitors and controls SCADA managers and OPC servers states in all DCS computers.

TRT DCS OPERATIONAL AUTOMATION

Software Safety Interlocks

There are two software interlock mechanisms implemented in the TRT DCS system. The LV interlock monitors temperatures measured on FE electronics boards. It switches off the FE LV in case of overheating. The HV interlock is triggered when Active Gas composition or its flow became not safe for straws (danger of discharges) or when humidity in racks housing HV power supplies became too high (danger for HV power supplies). It sequentially sets action flags (SAFE_HV, PREPARED, OFF) after programmable delays. All HV cells DU's call back functions perform required actions and block operator actions to set higher voltages until flags are active.

GGSS to HV Feedback

The GGSS provides reference HV and reference temperature for actual active gas mixture. Selected groups of temperature sensors are used to calculate mean temperature in different detector regions. Each 3 min. a timer triggers execution of a script which checks if GGSS references can be trusted and loops over all connected to detector HV cells performing the following steps:

- Checks if corresponding region temperature can be trusted.

- Calculates a new HV cell's set-point (basing on HV reference and temperatures difference between region and GGSS straws box).
- Checks if a new set-point value doesn't exceed acceptable range.
- Sends a new set-point value to HV cell, if all safety checks succeeded. If any of them fails, the GGSS to HV feedback is disabled till expert checks a situation and re-enables it.

The GGSS feedback corrects HV set-points for cells which are switched to state ON by FSM commands only, thus not interfering with manual operator or expert actions.

Cell Automatic Trip Recovery

The HV cells trips are usual events during the TRT detector operation and have to be carefully handled by experts who decide:

- which of the often tripping cells can be automatically switched back on,
- how many times the cell can be recovered (per day and since last counter reset),
- how long the automatic action has to be delayed.

After a delay HV cell's DU call back function automatically sends a sequence of commands to set a nominal voltage. Automatic trip recovery is performed when the trip happened from cell's ON state only.

LV Settings Equalisation

The distances between LVPP's and the detector FE boards range from 11 to 14 m. Also the current consumption of FE boards differs according to their function within the detector. There are in total 1792 FE boards in the system, each supplied with 3 voltages. Voltages levels at FE should be equal to nominal, so each voltage in the LVPP should be set to a unique value. Setting voltages levels is done via DACs (Digital-to-Analogue Converters) in the LVPP's. The setting is done after each longer shutdown period, board replacement or at any time, when there are some doubts on proper detector performance.

The Low Voltage Equalization Procedure has been prepared for this purpose. Its usage is foreseen not only for setting DAC values but also for debugging and checking the entire LV system in case of problems with operation of FE electronics. The Low Voltage Equalization Procedure consists of an external DAQ LV Equalization application calculating DAC values and mechanisms in PVSS project, including Framework components, which set these values to the hardware.

The DAQ LV Equalization application is performing calculations of DAC settings, and alarm limits for voltages and currents for each voltage type in individual FE board. Primary calculations are based on direct VT readout from the FE. The following, second way of calculations is for comparison and checks. Input parameters for this method are voltages and currents in the LVPP's and previous DAC settings, which are stored in the PVSS archive. These calculations are based on hardware topology, cables resistance and connection of grounds. Calculation results predict values of DAC settings and limits of alarms. They are stored in recipes in the Configuration DB and applied to the hardware from the PVSS project.

A PL/SQL API interface - consisting of a set of routines stored in the Oracle database server, being a part of the Configuration DB schema and allowing access to database data has been developed. These routines, called API Stored Procedures, are used by the DAQ LV Equalization application for fetching necessary data from the PVSS archive and storing output data to recipes in Configuration DB.

CONCLUSIONS

The TRT DCS is operating the detector very stably and reliably and is an important factor for detector safety, excellent performance and good quality of data taken during LHC collisions.

The hardware failures can't be avoided in such a big and complex system. Some of them can be recovered without stopping the ATLAS data taking, especially during LHC collisions, but a dedicated DCS tools for it need to be developed. HV cables „hot swap" to a spare (free) connector position tool is already in operation, the other will come in a near future.

ACKNOWLEDGEMENTS

We acknowledge the financial support of CERN; MNiSW, Poland; MES of Russia and ROSATOM, Russian Federation; DOE and NSF, United States of America.

REFERENCES

[1] Inner Detector Technical Design Report ATLAS TDR 4, CERN/LHCC/97-16 (1997).
[2] ETM professional control: PVSS, http://www.etm.at
[3] O. Holme et al. "The JCOP framework", ICALEPCS 2005, Geneva.
[4] C. Gaspar and B. Franek, Tools for the automation of large distributed control systems, IEEE Trans. Nucl. Sci. 53 (2006) 974.
[5] B. Hallgren et al. "The Embedded Local Monitor Board (ELMB) in the LHC front-end I/O control system", in Proc. 7th Workshop on Electronics for the LHC Experiments, Stockholm, Sweden, 2000.
[6] Gaspar, C., et al., Comput. Phys. Commun., vol 140, pp. 102-109, 2001.
[7] http://www.hvsys.dubna.ru/SC508_atl.pdf
http://www.hvsys.dubna.ru/CELL_ATL.pdf
[8] Z.Hajduk et al. "Distributed Low Voltage Power Supply System for Front End Electronics of the TRT Detector in ATLAS Experiment", in Proc. of the 12th Workshop on Electronics and Future Experiments, València, Spain, 2006.
[9] M. Deptuch et al. Gas Gain Stabilisation System prototype for the TRT - ATL-INDET-2002-010.
[10] Z.Hajduk et al. Gas Gain Stabilisation System - from model and prototype to final construction ATL-IT-ER-0025.

TURN-KEY APPLICATIONS FOR ACCELERATORS WITH LABVIEW-RADE

O. Ø. Andreassen, P. Bestmann, C. Charrondière, T. Feniet, J. Kuczerowski, M. Nybø, A. Rijllart,
CERN, Geneva, Switzerland

Abstract

In the accelerator domain there is a need of integrating industrial devices and creating control and monitoring applications in an easy and yet structured way. The LabVIEW-RADE framework provides the method and tools to implement these requirements and also provides the essential integration of these applications into the CERN controls infrastructure.

We present three examples of applications of different nature to show that the framework provides solutions at all three tiers of the control system, data access, process and supervision.

The first example is a remotely controlled alignment system for the LHC collimators. The collimator alignment will need to be checked periodically. Due to limited access for personnel, the instruments are mounted on a small train. The system is composed of a PXI crate housing the instrument interfaces and a PLC for the motor control. We report on the design, development and commissioning of the system.

The second application is the renovation of the PS beam spectrum analyzer where both hardware and software were renewed. The control application was ported from Windows to LabVIEW-Real Time. We describe the technique used for a full integration into the PS consoles.

The third example is a control and monitoring application for the "CLIC two beam test stand". The application accesses CERN front-end equipment through the CERN middleware, CMW, and provides many different ways to view the data. We conclude with an evaluation of the framework based on the three examples and indicate new areas of improvement and extension.

INTRODUCTION

At CERN there are different frameworks for application development, communication and front-end tasks programming. Adding specialized instruments that use dedicated software while making use of different frameworks can become a difficult and time consuming task. This led us to create the RADE framework a few years ago: a Rapid Application Development Environment realized in LabVIEW that covers all three control tiers. With the RADE framework developers and engineers don't have to cope with all the different interfaces and hardware and can focus on the job at hand, creating turnkey control and monitoring applications in a quick, stable and flexible way.

RADE

The RADE framework aims to give users a total package for development, maintenance and support, making it quick to implement flexible, stable and maintainable through well defined development templates, guidelines and documentation [1].

Templates, documentation, source control, updates, and libraries are all part of the framework, ready to be used in a few clicks.

The RADE distributed architecture provides a quick and flexible mean to interface with almost all equipment at CERN. Databases, frontends, file systems, PLC's and servers are all reachable through RADE. If a toolkit, protocol or library is missing it can easily be added to one of the framework's application servers or added on to the framework's native library package [2].

APPLICATIONS USING RADE

RADE can, with its high flexibility and wide range of libraries, be used in almost any control or monitoring application. Three applications covering different needs are shown here.

The LHC Collimator Survey Train - Multiple Alignment Control System (MACS)

The beam cleaning insertions in point 3 and 7 will become one of the most radioactive zones in the LHC. It was obvious since the beginning that standard alignment measurements will no longer be possible due to the high radiation level of up to 4mSv/h. The concerned zone is a 500 m long straight section of the LHC tunnel with 37 collimators and 26 reference magnets to be measured. The precision should be comparable to the rest of the LHC ring; meaning 0.15 mm in a sliding window of 200 m. Conventional measurements would take 4 days for a team of 3 people. The measurements must either be extremely reduced or one had to find a way to execute the measurements remotely controlled. Reducing the measurements is not considered as an option, as the collimation system is one of the most important parts of the machine protection system.

The basic strategy [3] was to use as much as possible off the shelves components. Various options have been studied, and finally a combination of two different techniques has been chosen. The system is based on a "MoveInspect" photogrammetric measurement system from AICON 3D Systems. This is a fast, precise and non-tactile way to measure the sockets of the radioactive collimators. This system is limited to a relatively small

volume and a considerable effort is needed to cover the 500 m of LHC tunnel.

In order to cover the whole zone, a stretched wire reference is used. More precisely, 5 overlapping and fix installed wires are used to connect the different acquisition volumes of the camera system. The aim is to measure the position of the collimators with respect to the surrounding reference quadrupole magnets at the extremities of the wires.

The software developed for the train project has been made using LabVIEW. The software is divided into two parts, the sequencer (Fig. 1) running on the train and communicating with instruments, and the remote GUI giving information to the user who will be positioned outside the tunnel. The application communicates with the PLC on the train using the "Fetch-Write" protocol from Siemens™ through TCP-IP. The other instruments in the systems (ie MoveInspect Camera system from AICON™, Wyler™ Zerotronic inclination sensors, Aeroel™ XLS35 laser micrometer) have been controlled using separate homemade drivers, communicating either through TCP or RS232 protocols.

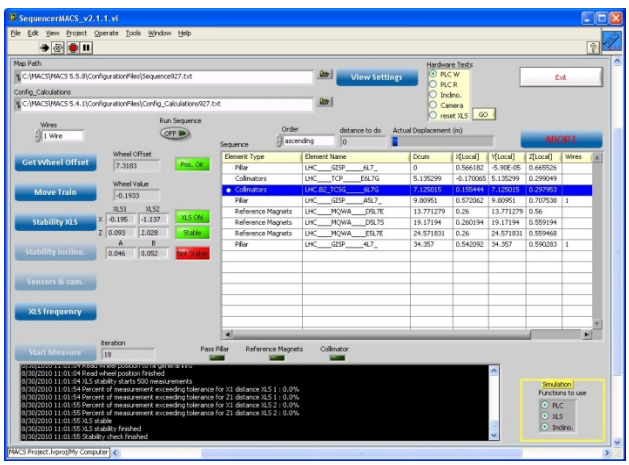


Figure 1: MACS Sequencer panel.

The sequencer is the main software and takes over the control of the whole train once activated. The remote GUI gives the operator the progress and status of all running applications, such as status of all sensors, results of stability tests, measurements and final results. In addition to the MACS software some other programs are used. The AICON MoveInspect software for the cameras, the CERN SU software Chaba for transformations and LGC++ are used finally for the calculation of the alignment.

Renovation of the PS Beam Spectrum Analyzer

The PS Beam Spectrum analyser was a single Windows based PXI crate from National Instruments, hosting dedicated RF cards used to perform Spectrum analysis on RF signals from the PS.

The analyser was mainly designed to perform spectrum and other frequency analysis on RF signals in the PS, but it is also used to do spectrum analysis on OASIS signals [4], which are obtained from remote hardware through a standard protocol.

Since this system is an integral part used to diagnose the RF signals in the PS, operators didn't think the application, running on Windows, was properly integrated into the CERN Control Room (CCC) consoles running Linux.

The PXI solution had so far worked fine, so it was decided that the new system would have a similar setup, but with the user application on the Linux console (Fig 2).

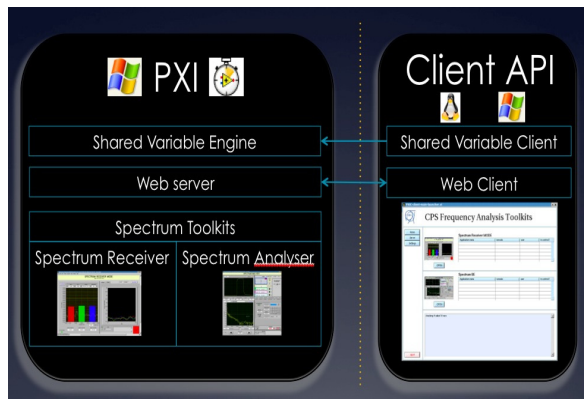


Figure 2: Spectrum Analyzer concept.

Two sets of PXI front-ends with a real time operating system called ETS and dedicated RF cards where installed and the existing software on Windows was converted into a split client-server application making use of LabVIEW remote panels and RADE [5].

The client part (Fig. 3) was adapted to the Linux system running on the consoles in the CCC and the server part was compiled for LabVIEW Real Time (ETS).

The communication between server and client was done using Shared Variables, a standard LabVIEW protocol, and the server would control its real time applications through a state machine based master.

This implementation concept proved quite powerful since any analysis application could later be added to the server and called from the client running in the CCC, and several applications still under development will be added at a later stage.

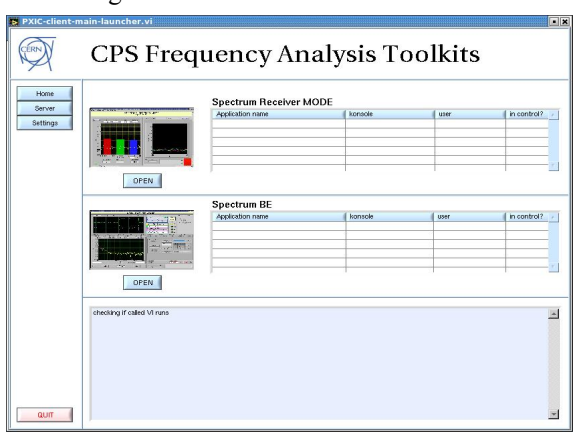


Figure 3: Spectrum Analyzer Linux Client.

With the use of LabVIEW and RADE the conversion of the old software and the installation of the new systems went quite smooth. A few adjustments to the real time headless environment had to be done.

Two-Beam Test Stand

The Two-Beam Test Stand (TBTS) in the CLIC experimental hall (CLEX) is aimed to test the two-beam acceleration scheme for CLIC, an electron-positron collider project. Beam stability, beam loss issues and RF structures are studied in this set-up.

All the signals are acquired through FESA devices, which are easily accessible using the CMW wrapper or the JAPC interface. The aim for this project is to be able to visualise the Beam Position Monitor (BPM) and RF signals in many different views such as horizontal vs. vertical graph or beam intensity or as a history graph. The BPM are probably the most important diagnostic devices used from the beginning.

First one needs to make sure that the beam actually gets to the dump and that it has roughly the same amplitude all the way. The RADE framework has been used to run the acquisition as a server task. Each graphical user interface then acts as a client taking only the necessary data to display in the appropriate format, without unnecessary data duplication.

The beam can be followed from the LINAC to TBTS within a single view (Fig. 4). When the beam is ready to be sent to the desired experimental area, the user can switch to a dedicated GUI to the see the beam position in time (Fig. 5).

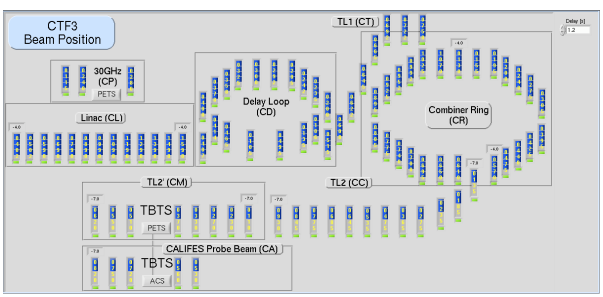


Figure 4: CTF3 beam position.

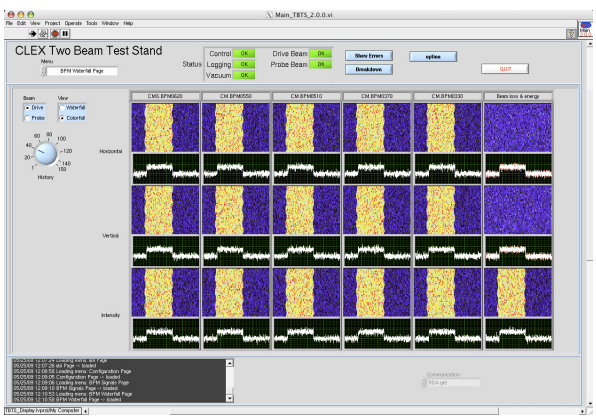


Figure 5: TBTS waterfall display.

The user can also adapt the current in the magnets directly from this application using RBAC and RADE [5]. Right clicking on the synoptic allows the operator to have access to magnet and valve parameters (Fig. 6).

Using the same scheme, a second application has been created for CALIFE experiment, still in the CLIC experimental hall (CLEX).

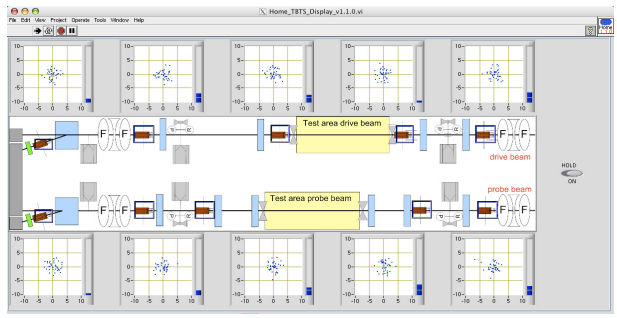


Figure 6: TBTS beam position and magnet controls.

CONCLUSION

The described examples above all use different technologies and techniques for communication and controls between software and equipment. The RADE framework and LabVIEW made it possible in a convenient and flexible way to realize the projects in a uniform environment within a reasonable time.

REFERENCES

[1] A. Raimondo, O. O. Andreassen, ”Rapid Application Development Environment based on LabVIEW”, CERN, Geneva, 2008,http://cern.ch/Rade.

[2] O.O. Andreassen “The LabVIEW RADE Framework” ICALEPCS 2011

[3] P. Bestmann, “The LHC collimator survey train”, IWAA 2010, DESY, Hamburg, Germany, September 2010

[4] The OASIS project, http://project-oasis.web.cern.ch/project-oasis/

[5] C. Charrondiere, “Controlling your PXI from the CCC”, CERN, Geneva. April 2011

WEB-BASED CONTROL APPLICATION USING WEBSOCKET

Y. Furukawa
SPring-8/JASRI, Kouto, Sayo-cho Hyogo, 679-5198, Japan.

Abstract

The WebSocket [1] allows asynchronous full-duplex communication between a Web-based (i.e. JavaScript-based) application and a Web-server. WebSocket started as a part of HTML5 standardization but has now been separated from HTML5 and has been developed independently. Using WebSocket, it becomes easy to develop platform independent presentation layer applications for accelerator and beamline control software. In addition, a Web browser is the only application program that needs to be installed on client computor. The WebSocket-based applications communicate with the WebSocket server using simple text-based messages, so WebSocket is applicable message-based control system like MADOCA, which was developed for the SPring-8 control system. A simple WebSocket server for the MADOCA control system and a simple motor control application were successfully made as a first trial of the WebSocket control application. Using Google-Chrome (version 13.0) on Debian/Linux and Windows 7, Opera (version 11.0) on Debian/Linux and Safari (version 5.0.3) on Mac OS X as clients, the motors can be controlled using a WebSocket-based Web-application. Diffractometer control application use in synchrotron radiation diffraction experiment was also developed.

INTRODUCTION

Web-based applications have many advantages over custom-built applications developed using Graphical User Interface (GUI) building toolkits: 1) platform independence; 2) easy to develop; 3) and easy to distribute the latest version. Based on these advantages, there have been many attempts to introduce Web-based applications for accelerator and experimental control.

Platform independence comes from the fact that most major Web browsers can run not only on traditional personal computer or workstation OSs such as Microsoft Windows, Mac OS X, most Linux distributions, Solaris, and FreeBSD, but can also run on mobile device OSs, such as Android OS and iOS. This gives us many candidates for an operator console and different styles of operation of accelerators and physical experiments. Using a mobile device, operators can access equipment besides it, and can easily troubleshoot the devices, if required.

There are many tools available for developing Web-based applications, including independent design and development tools such as Microsoft Visual Studio and Eclipse, and built-in browser tools. It is easier to develop applications using these tools than to use other language and tool kits.

A Web-based application is loaded from a Web server before every execution. It is easy to maintain the latest versions of all the running applications because you need to replace the Web application on the web server; you need not distribute and ask your application users to install a new version.

Despite these advantages, Web-based applications have a big disadvantage, i.e. a Web-application (HTML-based application) cannot make a full-duplex communications with a server application because the HTTP communication starts only from a client. This means that a server cannot send any information to a client without access from the client.

To solve this problem, a “long polling” mechanism was introduced in “Comet” [3]. A Comet server does not respond immediately to the client request (usually XHR is used). It waits for a server event, and the server then responds to the client with information on the event. Actually, the Comet server has to respond with in a definite time (usually 30 s) because Web browsers detect HTTP session time out.

This mechanism works well. However, it requires that an HTTP communication session be started for every long polling request. This results in a heavy; load on the Comet server and it is impossible to send information on client events to the Comet server during long polling periods. This means that it is difficult to realize a “true” control application using Comet.

The WebSocket [1] protocol was proposed to give true full-duplxy communication ability to Web-based applications in the context of HTML5. WebSocket is currently independent of HTML5 standardization, but is one of the important components of HTML5.

WebSocket is very easy to handle in a Web-based application, i.e., it can be easily handled in a program written using JavaScript. WebSocket can handle text and binary data. However, it is easy to handle text data in a JavaScript application. This means that it is easy to develop a WebSocket-based application for a text-message-based control system like MADOCA[6] or STARS[7]. Once you make a gateway from the WebSocket to your control system, you can make JavaScript-based applications. Therefore, a gateway (WebSocket server) to the MADOCA was built and several Web-based applications were tested, as described in the following sections.

SERVER APPLICATION

A schematic of a control system with WebSocket and MADOCA is shown in Fig.1. A WebSocket server has been newly built as a gateway to the MADOCA control system. The server has to treat the negotiation phase of

the WebSocket protocol. This protocol has been revised and the latest version is hybi-10[4] but most Web browsers are implemented on hybi-00[5]; therefore, our WebSocket server is implemented on the basis of hybi-00. The WebSocket server can co-exist with an HTTP server using the same port, i.e., 80 (or 443). In this case, the Web server has to treat the negotiation phase of the WebSocket, while our WebSocket server is running on port 10101 because of continuity of service. To control optical components in the SPring-8 X-ray beamlines, a request server is introduced for the beamline control system on the port 10101.

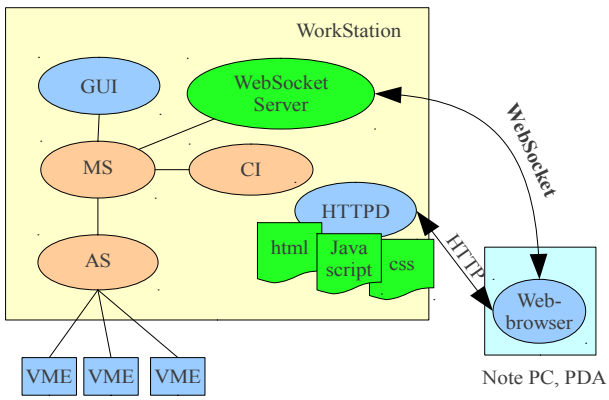


Figure 1: A schematic of WebSocket and MADOCA control system. Orange faced component, MS (Message Server), CI (Command Interpreter) and AS (Access Server) are standard MADOCA components [6]. Green faced components are newly developed for WebSocket Application.

This server is a simple socket server and with a small modification, we can adapt the server for both simple socket service and WebSocket service. The server decides which service is required from the first received string. For the simple socket service, the MADOCA format message will be received and for the WebSocket service, the format defined in hybi-00 or hybi-09 will be received, which is the same as an HTTP request, i.e., " GET /demo HTTP/1.1" or "GET /ws HTTP/1.1". After establishing a connection with a client, the WebSocket server receives MADOCA format from the client and sends it to a MADOCA-based control system. The WebSocket server then receives results from the MADOCA-based control system and sends them back to the client.

CLIENT APPLICATION

Client applications are written in JavaScript. In JavaScirpt, you can open the WebSocket using the "ws" object, as shown below

```
ws = WebSocket("ws://<hostname>[:port]");,
```

send a message with

```
ws.send(<message>);,
```

and receive a message with

```
ws.onmessage() = function(event) {
    var message = event.data;
    …
}.
```

Note that the receive routine is called asynchronously. So you need to handle it with an event-driven handler.

As shown above, the WebSocket interface can be handled very easily.

EXAMPLES

A stepper motor control GUI was developed for the first case of the WebSocket application as shown in Fig.2. The user can watch the motor status, can control motor rotation and can modify motor parameters using this Web page.

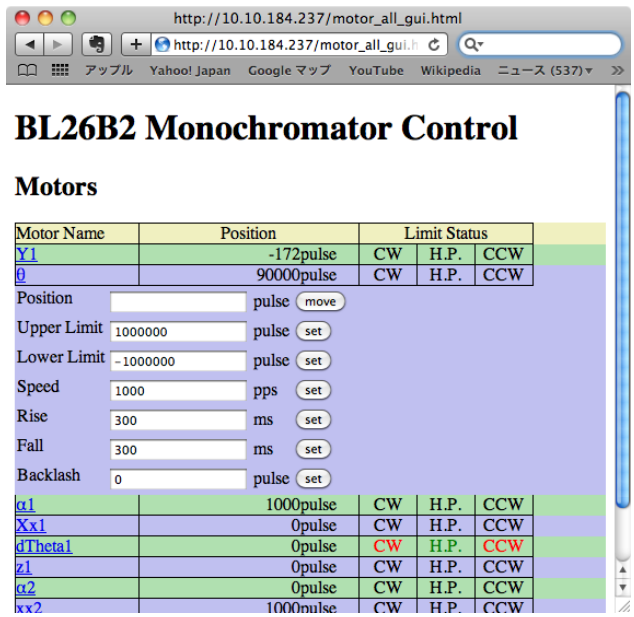


Figure 2: Web-application for stepper motor control.

A more complicated example shown in Fig. 3. is a user interface for a diffractometer control system at the SPring-8 diffuse scattering beamline. This Web page was designed using BlueGriffon™[8] which is a useful tool for building Web pages. Using this Web page, a diffractometer user can tune the slit and detector conditions and make a series of X-ray diffraction measurements. The scan results are shown in real time in the graph area which is realized by a "canvas" element defined in the HTML5 specification. In this case, the "RGraph" library [9] is used to draw the graph in the canvas element.

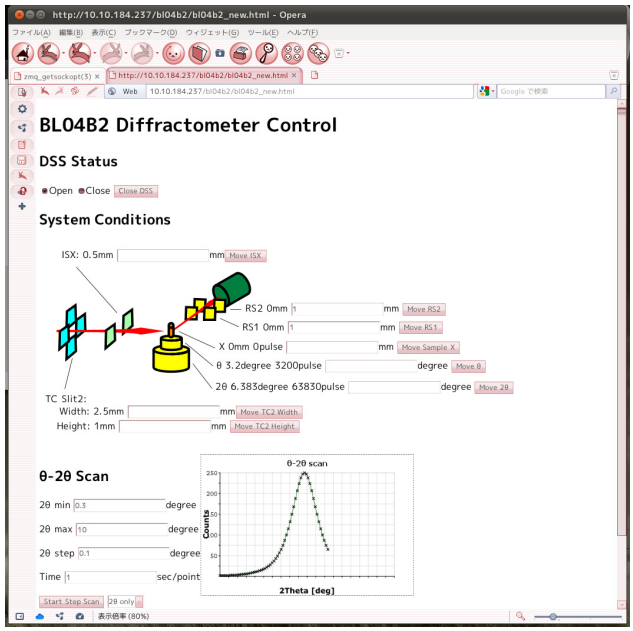


Figure 3: A diffractometer control Web-application.

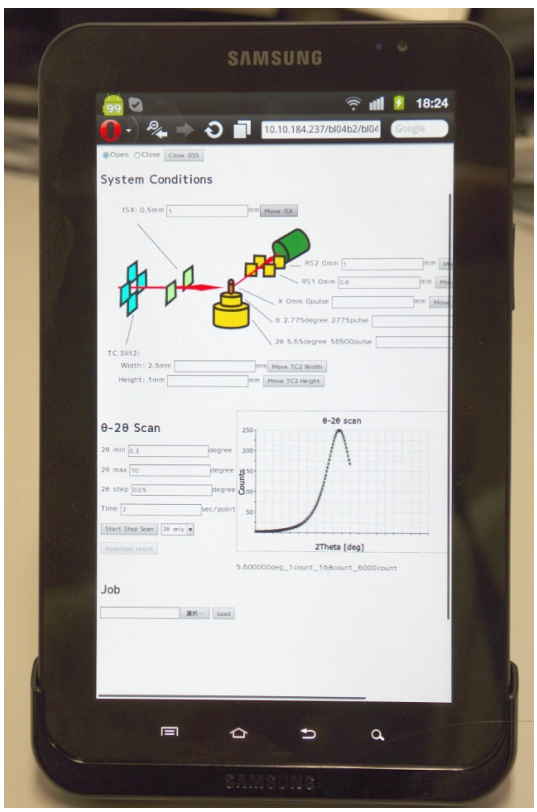


Figure 4: The diffractometer control Web-application running on a mobile device (SAMSUNG Galaxy Tab).

AVAILABLE WEB BROWSERS

The Following Web browsers were tested using the diffractometer-control GUI on a PC, and it was found to work well: Google-Chrome 13.0 on Windows, Debian/GNU Linux and Ubuntu 11.04; Safari on Mac OS X; Firefox 4 on Ubuntu 11.04; and Opera 11.50 on Debian/GNU Linux and Ubuntu 11.04. For Opera and Firefox, WebSocket is disabled as default, so you need to turn on it using "about:config."

Web browsers on mobile devices, Opera Mobile 11.10 on Android OS 2.3.3 (you need to enable WebSocket) and Safari on iPad (1st generation) were available for the WebSocket client. Fig. 4 shows a diffractometer control running on "SAMSUNG Galaxy Tab" which is an Android based tablet with a 7 in LCD screen.

CONCLUSION

As shown in this paper, Web-application can be built using the WebSocket protocol. I have to emphasize that no add-ons are required for executing WebSocket-based applications, just an HTML5 based Web browser. HTML5-based Web-applications provide freedom from the platform, so you can use Web-based applications not only in the central control room but also with various type of equipments. This means the Web-application can be used for local tuning or trouble shooting of the equipment. The web-based application is also useful for remote experiments. You can easily maintain the newest application running on the remote user's PC because the application is loaded from the Web server every time it has to be executed.

REFERENCES

[1] http://websocket.org/
[2] L. Zambon, M. Lonza, "WEB GUIS FOR THE TANGO CONTROL SYSTEM", Proc PCaPAC 2006 P.75 (JLab. USA).
[3] http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/
[4] http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-10
[5] http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-00
[6] R. Tanaka, T. Fukui, K. Kobayashi, T. Masuda A. Taketani, T. Wada and A. Yamashita, "The first operation of control system at the SPring-8 storage ring", Proc. of ICALEPCS'97, Beijing, China, (1997) p.1
[7] T.Kosuge, et., al., "Recent Progress of STARS", Proc PCaPAC 2005 (2005, Hayama, Japan)
[8] http://bluegriffon.org/
[9] http://www.rgraph.net/

LIMA: A GENERIC LIBRARY FOR HIGH THROUGHPUT IMAGE ACQUISITION

A. Homs#, L. Claustre, A. Kirov, E. Papillon, S. Petitdemange, ESRF, Grenoble, France.

Abstract

A significant number of 2D detectors are used in large scale facilities' control systems for quantitative data analysis. Common control parameters and features can be identified in these devices, but most of manufacturers provide specific software control interfaces. A generic image acquisition library, called LIMA, has been developed at the ESRF for a better compatibility and easier integration of 2D detectors to existing control systems. Its design is driven by three main goals: i) control system independence; ii) a rich common set of functionalities, providing alternative software solutions when not implemented by hardware; and iii) intensive use of multi-threaded algorithms synchronised by events for ensuring performance of high throughput detectors. LIMA currently supports a dozen of image detectors from different manufacturers, half of them being integrated by collaborating institutes and manufacturers. Although still under development, LIMA features so far basic image transformation, processing and reduction, fast data saving on different file formats, among other features. Its modular design allows not only the extension of generic hardware and software functional blocks, but also the integration of user-defined post-processing algorithms.

INTRODUCTION

The standard beamline (BL) control software at the ESRF is basically built on top of SPEC, a versatile hardware control application, and a variety of distributed device servers accessed through the TACO and TANGO middlewares. A considerable number of 2D detectors have been interfaced in the past using common generic CCD TACO and TANGO device server interfaces, notably simplifying the client SPEC code. However, the real code implementing the configuration and control of the image acquisition process was copied and/or rewritten on each new detector. Moreover, new features had to be manually added on existing detectors, which resulted unpractical in many cases.

In order to go a step further in the standardisation of the control of these devices, we have developed a generic Library for IMage Acquisition (LIMA) [1]. It provides the common functionality expected from 2D detectors, exploiting the hardware capabilities to their maximum extent, and complementing by software the rest features not provided by the detector.

BUILDING BLOCKS

A key design goal was to not degrade the performance of high throughput detectors. This implied the use of parallelisable multi-threaded algorithms for an optimum usage of multi-CPU/cores in modern PCs, and their corresponding event-based synchronisation. The result was the development of ProcessLib, an auxiliary multi-threaded image processing framework. It manages the execution of arbitrary sequences of tasks, combined in chained and/or parallel configurations. Once added to the run queue, a task is executed as soon as a working thread is available, normally when a CPU/core is idle. The asynchronous notification of the task end is implemented through callbacks.

Another basic element developed for LIMA is a message logging framework for code tracing and debugging. It can be dynamically configured to enable/disable individual functional blocks, and, in an independent way, to change the log verbosity. The output trace is indented following the function call stack, which is different for each active thread. Special efforts were made to reduce code execution slowdown when no trace is active, while keeping this dynamic, thread-safe behaviour.

The core of the library has been written in C++ and Python wrapping through SIP has been implemented for most of the classes.

LIBRARY STRUCTURE

The general LIMA layout is shown in Fig. 1.

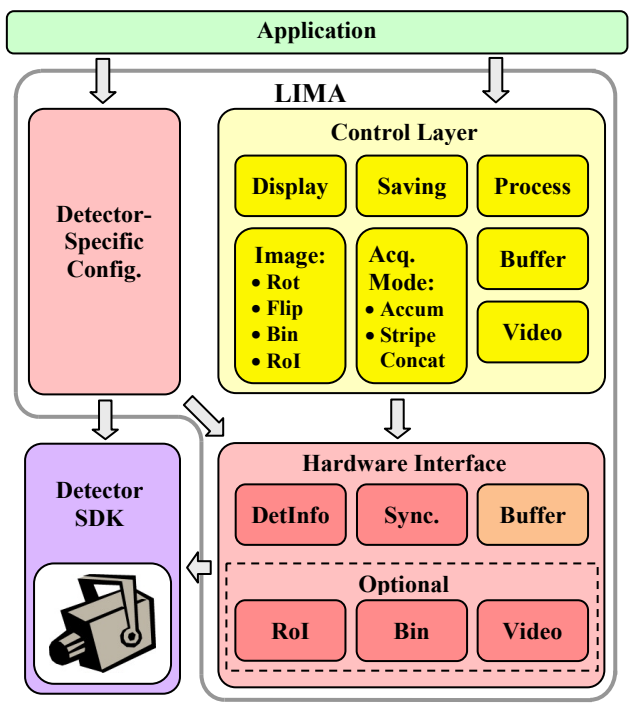


Figure 1: General LIMA layout.

#alejandro.homs@esrf.fr

The library structure divided into two main layers: control, containing the common control and processing code, and hardware, implementing the detector-specific part. The control layer provides the library interface to the high level application. User requests to configure and control the acquisition are gathered by the control layer, so the hardware layer functionality is limited to the generation the image frames in a best-effort basis. The control layer is responsible of: i) adapting the received image geometry if it does not match the user requests, and ii) execute the frame processing chain.

Hardware Layer

The detector-specific code, called "hardware plug-in" must implement a well-defined "hardware interface". It provides basic acquisition start/stop/status control as well as entry points to functional blocks, corresponding to the hardware capabilities. Only three control blocks are mandatory:

- Detector information: name, model, dimensions.
- Synchronisation: internal/external trigger, exposure time and sequencing.
- Image buffer management, for which helper classes are provided with a default behaviour.

All the other capabilities are optional. They include:

- Pixel binning (scaling).
- Sub-image / region-of-interest (RoI).
- Image horizontal / vertical flip.
- Shutter control.
- Video cameras: standard mono/color image formats and gain settings.

This modular design simplifies the integration of new hardware functionality without recoding existing hardware plug-ins.

One important requirement imposed on the hardware plug-in is its responsibility of generating events on every acquired frame. In case the detector API does not provide this feature and relies on blocking/polling interfaces, auxiliary thread management helper classes are provided to simplify the implementation of frame ready callbacks.

As a general rule, the hardware layer is queried to check if it can completely or partially satisfy a particular user request. For instance, when the user asks for a pixel binning combination that is not implemented, the hardware layer replies with the highest binning it can provide. The same best-effort approach should be followed with RoIs, where some hardware alignment constrains might require returning a region bigger than the requested one. The user can always select if a feature is to be implemented entirely by hardware, by software or by a combination of both.

It is worth to remark that to add support for a new detector its only required to implement:

1. The detector information control block
2. The synchronisation control block
3. Acquisition start/stop/status control and frame-ready callbacks

This can be coded either in C++ or Python. Such a minimal implementation will already provide the user with a fully operational system exporting all the common software features.

Control Layer

A modular approach is also found in the control layer, the core of LIMA and its most complex element. Different functions are activated through individual blocks, all of them coordinated by a global control class. The user configures the acquisition using these different modules, many of them implementing parameter caches to minimise hardware access. Once all the settings have been fixed, a "prepare acquisition" command synchronises the hardware configuration, including the (sometimes slow) memory buffer allocation. This minimises the delay in the "start acquisition" command, which can add undesirable latencies in the software synchronisation with other devices like motors or LIMA detectors.

The dynamics of the control layer starts with the "hardware frame ready" callback. The first task to accomplish is the software frame reconstruction, needed if the detector pixel readout sequence does not correspond to its real geometry [2]. Even if this is a detector-specific functionality, it is performed in the control layer to exploit the ProcessLib capability of parallel processing different frames on different CPU/cores.

Individual counters are updated as the frame progresses in the processing chain: last-image-acquired (hardware), last-base-image-ready (geometric reconstruction and transformations), last-image-ready (basic processing), last-image-saved (data storage), etc. The user can register to an acquisition status callback that notifies each time these image counters progress.

Detector-specific Configuration

The control layer provides a user interface to standard acquisition parameters. However, virtually all detectors implement specific settings, which can range from chip timing and readout configurations to generic I/O signal management and ADC gain/threshold levels. To avoid a complex generic infrastructure for these controls, the hardware plug-in is responsible to directly export them to the user through a detector-specific control block. The control layer is not aware of such parameters, so mechanisms have been foreseen to notify important changes that affect the acquisition. For instance, a "max image size changed" callback must be implemented by the hardware if the user can select among detector profiles with different effective image sizes.

AVAILABLE FEATURES

Geometric Transformations

Four basic image transformations are implemented: image rotation (90º, 180º, 270º), horizontal and/or vertical flip, pixel binning (scaling) and RoI (sub-image). From the user coordinates point of view, they are applied in that

order. This means that binning factors include the rotation mode, and RoI coordinates are relative to binned pixels. The simultaneous activation of these transformations requires more complex calculations when some of them are either totally or partially done by hardware. In addition, the implementation of arbitrary pixel binning combinations, including values that are not integer divisors of the detector size, is under development.

Some scientific techniques exploit 2D detectors to measure 1D spectra in image stripes. Pixel binning in one dimension is normally used to improve signal to noise ratio. Such configurations can lead to a very high frame rate (above 1 KHz), interesting in time-resolved experiments [3]. To keep performance under these conditions, LIMA implements the stripe concatenation mode, where a long sequence of many frames can be read/saved at once as a single concatenated image without additional memory copies. In case the original 2D images are required for analysis (i.e., no binning applied), the same stripe calculation can be obtained by the independent RoI-to-spectrum software operation.

Basic Image Processing

Hardware detector constrains can limit the pixel integration either in exposure time (Frelon) or in dose (Maxipix [2]). A simple solution to this problem is provided through the frame software accumulation. It is activated by specifying a maximum frame exposure time; the control layer programs the real number of hardware frames depending of the requested total exposure. In addition to the accumulated image, a saturation pixel mask can also be obtained from the algorithm, important to detect non-linear software artefacts.

Basic image processing algorithms like background subtraction, flat-field correction and pixel masking are already integrated. Standard calculations used for Beam-Position-Monitoring (BPM) including intensity sum, average, std. deviation, min/max and centroid position are also available in a per-RoI basis. That is, the so-called "RoI-counters" are calculated in parallel on multiple sub-images for each frame. The history of the all the RoI-counters is available during and/or after a sequence of images (scan).

Data Saving

Data storage is a key element in high performance image acquisitions. In addition to the detector images, meta-data describing the acquisition environment, including user-supplied meta-data, must be saved. The meta-data concept is called "frame-header" in LIMA and is implemented in the core of ProcessLib as key→value maps. Three levels of meta-data are identified:

- Static: does not change during the life of the process (detector-specific: model, serial number).
- Common: is shared by all the frames in an acquisition sequence (user-defined sample name, sequence start date/time).
- Frame: specific information when frame was taken (high resolution time stamp, instantaneous internal and/or external counters values).

There are three file saving modes currently implemented in LIMA. They differ in the way to trigger the saving of each frame: manual (user request), auto-frame (frame is ready) and auto-header (both the frame and its user-defined header are ready).

The following file formats are currently supported:

- EDF: ESRF Data Format
- Nexus/HDF5: part of the Common Data Model (CDM), developed by SOLEIL and ANSTO
- CBF: Crystallographic Binary Files. It is optimised in LIMA with parallel frame compression.
- Raw

The library also allows the parallel saving of multiple file streams on different medias. This data replication technique has proven to be efficient in high throughput detectors when multiple endpoints (PC, file servers) must receive the same data. Data endpoints examples include: online data analysis workstation, central data storage server (backup, long-term archiving) and local NAS with BL user disks (to be brought to their home institute).

Online Visualisation

Online data visualization is required in must of the cases as a direct feedback of the acquisition evolution. Current LIMA display mechanisms export the image data to a channel, to be read by a separate client application that performs the real visualisation. One method publishes the data on the standard ESRF SPEC Shared memory (SPS). A second method exports the generic LIMA video interface through the TANGO LIMA server. In each case, a dedicated client Qub/Qt4-based application shows the live image.

External Software Plug-ins

Finally, user-defined software plug-ins can be used to execute arbitrary image-based operations. An entry point in the control layer completely exports the ProcessLib functionality, allowing an external code to be called on every frame. Again, the external software operation can be implemented C++ or Python.

PROJECT STATUS

Supported Detectors

The list of currently supported detectors is:

- Camera simulator: a pure software plug-in
- ESRF Espia-based Frelon camera and Maxipix detectors (Single chip, 2x2 and 5x1 chip arrays)
- Dectris Pilatus and Mythen
- GigE Basler and Prosilica cameras
- ADSC and MarCCD detectors
- XPAD pixel detector
- Roper Scientific cameras through PVCAM
- PCO.Dimax (under development)

Current Applications

The first application on top of LIMA was the ESRF Frelon TACO device server; aimed to be 100 % compatible with its predecessor. The existence of an important amount of SPEC code for different Frelon-based experimental setups allowed the fast commissioning and validation of the basic LIMA elements.

In parallel, a generic LIMA TANGO device server was developed, with the ESRF Maxipix detector as its first specific hardware plug-in. As new hardware plug-ins were added, this new TANGO interface gradually exported the full LIMA potential: multi-chip reconstruction, RoI-counters, frame accumulation, background/flat-field, video and so on. New SPEC code has been written to efficiently exploit this new 2D detector functionality.

A different YAT-based TANGO server interface has been developed and maintained by SOLEIL Synchrotron, allowing LIMA integration in their control system.

Finally, an X-BPM TANGO server on top of LIMA has been developed; its installation currently uses Basler cameras.

Operating Systems

LIMA has been mainly developed on Linux operating systems, mainly on SuSE Linux 8.2 (i686) and on RedHat EL 5 (x86_64). It has been later ported to RedHat EL 4 (i686) and OpenSuSE 11.2 (i686 & x86_64).

A Windows portage has been made by coding basic thread management primitives. After the validation of the camera simulator, real hardware plug-ins like the PCO.Dimax are under development.

Performance

The first validation of LIMA performance has been made with the ESRF Frelon HD camera controlled by a Transtec X2100 Workstation (Dual Intel Xeon @ 2.66 GHz, 32-bit). Continuous Frame-Transfer-Mode acquisitions at 30 frames/sec (120 MBytes/sec) can be performed with active SPS online display and parallel data saving. Higher frame rates (about 1000 frames/sec) can be achieved on the same PC in stripe (spectrum) mode with hardware binning and Kinetics hardware RoI. As with its non-Lima predecessor, debugging output must be disable to not slowdown acquisition.

The most recent high performance LIMA validation has been done at the ID03 ESRF BL with the Maxipix detector in single-chip configuration. The control PC is a more powerful Ecrin/Trenton server (Dual Quad Core Xeon E5335 @ 2.0 GHz). Acquisitions at the maximum frame rate (about 1500 frames/sec), limited only by the maximum Espia board data rate (180 MBytes/sec), are possible with parallel saving and 5 active RoI-counters definitions.

Code Repository

The LIMA source code is available under the Gnu Public License at the European Phonton-Neutron (EPN) Science Campus Web site [1]. Its development is managed by GIT; the repository can be accessed at [4]. The official project documentation can be found at [5].

ACKNOWLEDGEMENTS

Important contributions in hardware plug-ins and Nexus file saving format have been made by F. Langlois, A. Noureddine, A. Buteau and X. Elattaoui from SOLEIL Synchrotron inside a collaboration framework on LIMA development. In the same way, M. T. Nuñez-Pardo-de-Vera (PETRA III/DESSY) and C. Nielsen (ADSC) have also implemented specific hardware plug-ins.

CONCLUSIONS

A generic Library for IMage Acquisition (LIMA) has been developed for controlling high throughput 2D detectors. It allows the optimum exploitation of hardware optimizations, like pixel binning and RoIs, but it also provides software alternatives for detectors that do not implement them. A common set of software image operations (geometric transformations, processing, calculations, saving and visualization) is available for all the detectors. Their multi-threaded nature notably increase acquisition performance on multi CPU/cores PC. The modular library design simplifies the integration of new hardware and software functionality (plug-ins). An increasing number of supported detectors are already used in Synchrotron facilities with good performance results.

REFERENCES

[1] http://forge.epn-campus.eu/projects/lima
[2] C. Ponchut, J.M. Rigal, J. Clement, E. Papillon, A. Homs and S. Petitdemange, "MAXIPIX, a Fast Readout Photon-Counting X-Ray Area Detector for Synchrotron Applications", JINST 6 (2011) C01069.
[3] J.C. Labiche, O. Mathon, S. Pascarelli, M.A. Newton, G.G. Ferre, C. Curfs, G. Vaughan, A. Homs and D. Fernandez-Carreiras, "The Fast Readout Low Noise Camera as a Versatile X-Ray Detector for Time Resolved Dispersive Extended X-Ray Absorption Fine Structure and Diffraction Studies of Dynamic Problems in Materials Science, Chemistry and Catalysis", Rev. Sci. Instrum. 78/9 (2007) 091301.
[4] git://git.epn-campus.eu/repositories/Lima
[5] http://lima.blissgarden.org/

COMETE: A MULTI DATA SOURCE ORIENTED GRAPHICAL FRAMEWORK

G. Viguier, K. Saintin, Y. Huriez, M. Ounsy
SOLEIL Synchrotron, Gif-sur-Yvette, France.
R. Girardot, EXTIA, Sèvres, France

Abstract

Modern beamlines at SOLEIL need to browse a large amount of scientific data through multiple sources that can be scientific measurement data files, databases or TANGO control systems.

We created the COMETE[1] framework because we thought it was necessary for the end users to use the same collection of widgets for all the different data sources to be accessed. Conversely, for GUI application developers, the complexity of data source handling had to be hidden. As these two requirements are now fulfilled, our development team is able to build high quality, modular and reusable scientific oriented GUI software with consistent look and feel for end users.

CONTEXT

The SOLEIL ICA team is in charge of the control system and data reduction software on the beamlines. In the early years of development at SOLEIL, ICA has focused on the control system; today, however, the NeXus storage system is fully integrated into the control system through a set of dedicated devices. It is therefore possible to record any data coming from Tango devices into a NeXus file; SOLEIL beamlines produce daily experimental data files with sizes ranging from a few MB up to 100 GB. This is why there is a huge demand for data reduction applications.

The ATK[2] toolkit helped us to quickly develop applications dedicated to the control system and its supervision. The problem is that ATK is intimately linked to TANGO.

Data reduction software on the other hand is based on the experimental data NeXus file format and doesn't have a dedicated graphical toolkit. It is in this context that SOLEIL launched the COMETE project to propose a multi data source toolkit to help engineers to develop applications independently of the data source.

THE COMETE SOLUTION

Architecture

COMETE is a framework composed of three parts (Fig.1):

1. A set of graphical components (widgets) that are completely dissociated from a data source or even a data type.
2. A data source compatible with the graphical component, each corresponding to a data type.
3. Between the widgets and the data source, a communication layer called DataConnection Management.

Each component of this architecture will be described in the following sections.

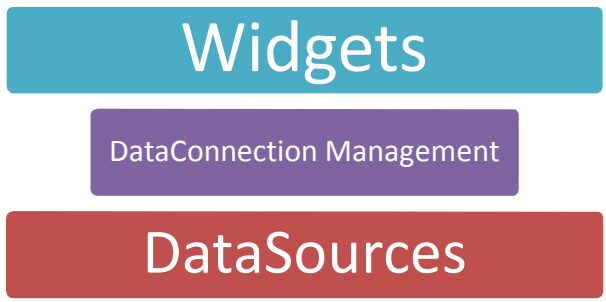


Figure 1: Comete Project Architecture.

Widgets

Comete Widgets are available in three implementations (Swing, SWT & AWT) and therefore integrate well into any existing JAVA software (Fig. 2). Anyone can use this set of widgets because they are based on the three major toolkits. In fact, a Comete Widget is a standard component with some code that makes it connectable to any COMETE data source.

COMETE components are dedicated to scientific data but anyone can easily add new widgets or use the existing components in another context.

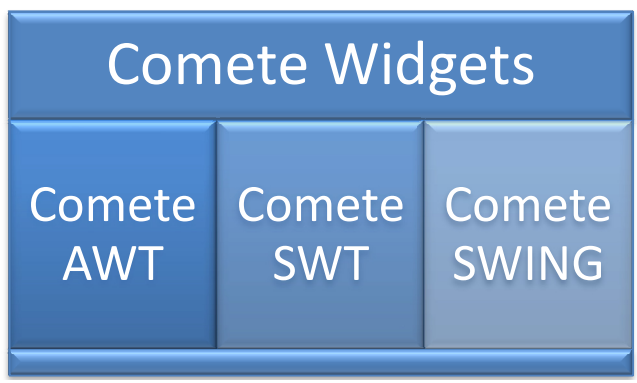


Figure 2: Comete Widget implementations.

The current widget can display:

- scalar data (textfield, spinner, wheelswitch, slider, etc.)
- spectrum data (chart viewer)
- images

Our image viewer is based on ImageJ [3] which is a popular image processing application. ImageJ is already used by many scientists, so it is very easy for them to use our viewer and they can run their old macros through our Java applications. Since our viewer encapsulates ImageJ, it has the same amount of image processing features as illustrated in Fig. 3.

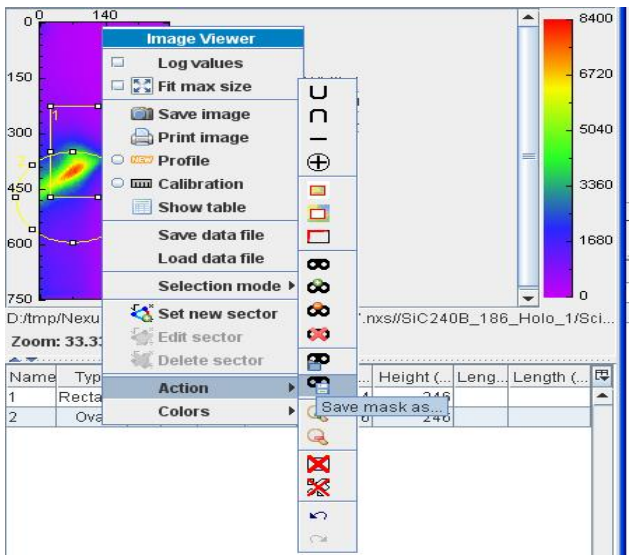


Figure 3: Comete Image Viewer features.

DataConnection Management

The DataConnectionManagement module is a layer that allows connection between two entities, called "target" and "data container".

This system implements a Mediator pattern, as well as various other patterns such as Strategy, Observer etc. Mediator was chosen instead of MVC pattern because our two entities had to be completely independent from each other and from the system evolution.

The focus for this project is to provide COMETE with a solution to connect a widget with a data source, like Tango or NeXus, in a generic way. That means this system is an abstraction of these entities, and provides useful tools to make them communicate without knowing their type.

For example, thanks to the Comete architecture, anyone can superimpose a spectrum coming from a NeXus file with the same spectrum from the control system (see Figure 4).

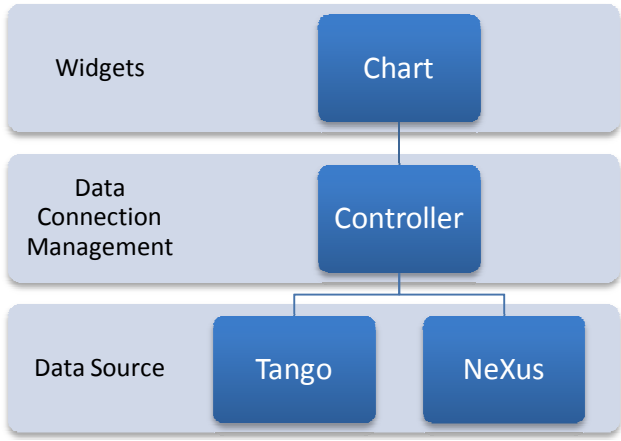


Figure 4: A widget connected to two sources.

DataSources

When someone wants to add a new data source to Comete, the only thing to do is to implement a class called AbstractCometeDataSource. This will make sure that the COMETE controller will be able to send data to the widget and vice-versa.

We made a TANGO implementation for control system software and we are working on a NeXus implementation for data reduction software. We plan to implement a data source for databases because we have an archiving system[4] built on ORACLE systems. Finally, we will have a data source for our sequencer software (Passerelle[5]) and thus enable our users to modify their sequences through our Java software.

CometeBox

The CometeBox module aims to simplify the use of COMETE. The fact that COMETE is a group of six different modules and the complexity of the architecture, especially the widget / source connection, makes it in fact difficult to use.

CometeBox operates on two levels. The first is for connection management: ordinarily, we need about ten to fifteen lines of code to make connections properly between entities, calling numerous functions coming from interconnected projects. With CometeBox, this is reduced to one line: one call to the correct function (Fig. 5).

The second goal of CometeBox is to centralize data handling: the connection process needs a wide range of data, some of them holding the same information. With CometeBox, this data is gathered and only information that is really necessary is stored.

NeXus

- NexusKey key = new NexusKey();
- NexusKeyTool.registerAttribute(key, "myfile", "dataset);
- chartBox.connectWidget(chartViewer, key);

Tango

- TangoKey key = new TangoKey();
- TangoKeyTool.registerAttribute(key, "mydevice", "attribute");
- chartBox.connectWidget(chartViewer, key);

Figure 5: Connect a widget to a data source in 3 lines.

Performance

Five years ago, SOLEIL scientists were working with spectrum detectors or small image cameras. Performance was not a big deal because the data flow was not really impressive. Today this is not true, scientists are buying fast acquisition detectors with big CCDs and they store each frame into NeXus files.

COMETE needs to have a fast and efficient mechanism to transfer data from the source toward the widget. The first version of COMETE was really slow because it was only based on Java objects (*for example: each integer transferred was converted into an Integer Java object*). We thought that this would make COMETE a modern Framework but the memory footprint of every COMETE data reduction program was too expensive. Then we changed our philosophy from 'all object' to 'no object'. That means that today, each data item is transferred through COMETE as a primitive type. Today we are able to load stacks of images from our high data throughput multi detectors experiments [6] in a reasonable time.

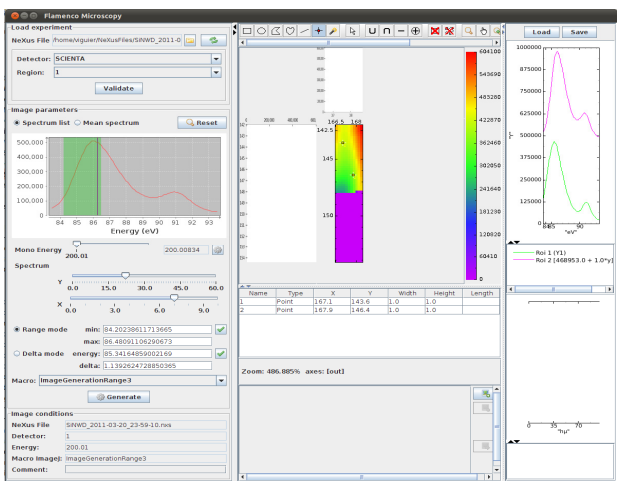


Figure 6: Data reduction software dedicated to microscopy.

EXAMPLES

Fig. 6 and Fig. 7 are two examples of graphical applications based on COMETE and using the same component list with different data sources.

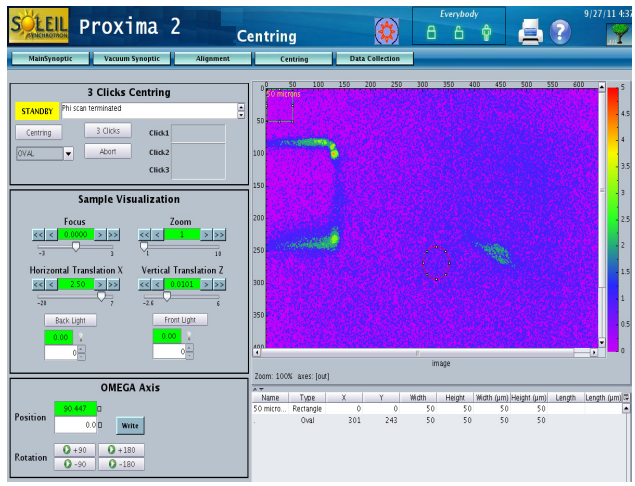


Figure 7: MXCube clone on PX2 beamline.

CONCLUSION

Today COMETE is a great tool for SOLEIL's developers. NeXus novices are able to build smart and effective data reduction software; anyone can develop a new supervision application without the help of TANGO gurus. In the future, we plan to optimize COMETE by implementing new refreshing systems. Data Reduction developers are looking forward to another feature: a mechanism of filters between the data source and the widget. Multiple filters can then be stacked and applied in a given order to an original Data Source to transform it to the final Data Source that will be really viewed by the COMETE widget.

REFERENCES

[1] G.Viguier, K.Saintin http://sourceforge.net/apps/trac/comete/
[2] F. Poncet, J.L. Pons, "Tango Application Toolkit", ICALEPS'05, Geneva, Oct 2005.
[3] ImageJ : Image processing and analysis in Java http://rsb.info.nih.gov/ij/
[4] J. Guyot, M. Ounsy, S. Pierre-Joseph Zephir "Status of the TANGO Archiving System", ICALEPS 2007
[5] A. Buteau, M. Ounsy, G. Abeille "A Graphical Sequencer for SOLEIL Beamline Acquisitions", ICALEPS'07, Knoxville, Tennessee - USA, Oct 2007.
[6] "Distributed Fast Acquisitions System for Multi Detector Experiments", ICALEPCS 2011, WEPKN003

FLOATING-POINT-BASED HARDWARE ACCELATOR OF A BEAM PHASE-MAGNITUDE DETECTOR AND FILTER FOR A BEAM PHASE CONTROL SYSTEM IN A HEAVY-ION SYNCHROTRON APPLICATION*

Faizal Arya Samman, Pongyupinpanich Surapong, Christopher Spies and Manfred Glesner
FG Mikroelektronische Systeme, Technische Universität Darmstadt, Darmstadt, Germany

Abstract

A hardware implementation of an adaptive phase and magnitude detector and filter of a beam-phase control system in a heavy ion synchrotron application is presented in this paper. The main components of the hardware are adaptive LMS filters and phase and magnitude detectors. The phase detectors are implemented by using a CORDIC algorithm based on 32-bit binary floating-point arithmetic data formats. The floating-point-based hardware is designed to improve the precision of the past hardware implementation that were based on fixed-point arithmetics. The hardware of the detector and the adaptive LMS filter have been implemented on a programmable logic device (FPGA) for hardware acceleration purpose. The ideal Matlab/Simulink model of the hardware and the VHDL model of the adaptive LMS filter and the phase and magnitude detector are compared. The comparison result shows that the output signal of the floating-point based adaptive FIR filter as well as the phase and magnitude detector agree with the expected output signal of the ideal Matlab/Simulink model.

INTRODUCTION

In a synchrotron, different modes of coherent longitudinal beam oscillations may occur due to an initial mismatch or to non-linearities such as wake fields. These oscillations are characterized by their mode number m and n [4] and take place at the characteristic synchrotron frequency, which depends on the system state (more precisely, on the magnetic flux derivative, accelerating voltage, and particle energy). In order to eliminate undesired dipole oscillations, a beam phase control system [5] has been devised, which was initially designed to deal with in-phase dipole oscillations ($m = 1, n = 0$) only [5]. The addition of amplitude detectors is intended to make it suitable for damping higher-order modes [4, 6].

This paper presents a new floating-point based architecture designed to improve both the precision and processing speed of the digital controller over previous implementations [2, 11]. The floating-point arithmetic units are used to overcome issues with explosive divergence and stalling effects caused by fixed-point implementations [8]. Such problems may arise due to input or output signal saturation caused by changes in the dynamic signal range at runtime. Using a floating-point representation eliminates the need for input and output scaling that would have to be performed in a fixed-point implementation to accomodate these changes. A floating-point implementation can also be adapted to different dynamic range requirements of other control applications and is therefore more suitable for an integrated circuit implementation (ASIC).

In the following section, the digital signal processing chain of the beam phase control is presented. Its core components are phase and magnitude detectors and adaptive filter blocks, which are discussed in the subsequent sections.

BEAM PHASE CONTROL

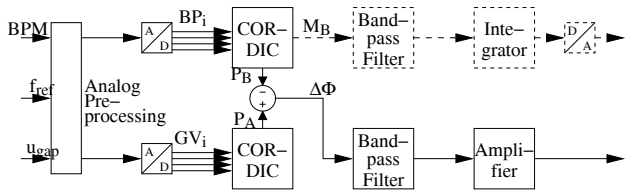


Figure 1: Block Diagram of the Beam Phase Signal Processing.

The signal processing pipeline implemented in the DSP system is shown in Fig. 1. It performs the following steps:

1. Some analog preprocessing is performed on both the gap voltage and the beam position signal [3]. Note that the beam position signal is quite noisy and has a high level of uncertainty since the bunch may have an arbitrary shape.

2. Four successive samples are used to form the in-phase and quadrature components of both signals.

3. The phase of both signals is computed using Alg. 3.

4. The phase difference is fed into a programmable FIR filter in order to eliminate noise and other disturbances in the phase difference. The filter is tuned to a frequency slightly above [5] the synchrotron frequency and has to be re-tuned continuously since this frequency changes.

5. The output of the filter is fed into a variable-gain controller whose gain also depends on the characteristic frequency. This controller yields a frequency correction which is then applied to the cavity in order to intentionally mistune it, thereby pulling the bunch back to its desired position.

BPM is the signal from the beam position monitor. u_{gap} is the voltage across the gap of the reference cavity. f_{ref} is the reference frequency (see [3]). Future extensions [6] are

* The work is supported by German Federal Ministry of Education and Research in the frame of Project FAIR (Facility for Antiproton and Ion Research), Grant Number 06DA9028I.

dashed. In order to enable real-time processing, the phase detector and filtering blocks have been implemented on an FPGA.

PHASE-MAGNITUDE COMPUTATION

Architecture for Phase-Magnitude Computing

The phase and magnitude of the beam signal are detected by using a CORDIC (COordinate Rotation DIgital Computer) algorithm. The drawback of a conventional CORDIC algorithm is its high computation latency [7]. To accelerate the computation, an increment of rotation angle on each iteration becomes one solution. A double-rotation CORDIC algorithm is proposed as depicted in Alg. 1. In the double-rotation CORDIC algorithm, the rotation angle is extended two times (2ϕ) of the conventional CORDIC.

Algorithm 1 MICRO-DRCORDIC-VM

Input: : $X_i, Y_i, Z_i, T, \delta_i$
Output: : X, Y, Z, δ_{i+1}
$X = X_i - 2^{-2\cdot i-2} \cdot X_i - \delta_i \cdot 2^{-i} \cdot Y_i$
$Y = Y_i - 2^{-2\cdot i-2} \cdot Y_i + \delta_i \cdot 2^{-i} \cdot X_i$
$Z = Z_i - \delta_i \cdot 2 \cdot T$

The double-rotation CORDIC algorithm using the redundant method was proposed by Takagi et al. [13]. With the redundant method, the rotation direction of the double-rotation CORDIC algorithm is $\delta_i \in \{-1, 0, 1\}$. However, using the redundant method, the scaling factor is not constant and has to be calculated at run-time. To overcome this problem, we propose the double-rotation CORDIC algorithm as presented in Alg. 2 using the non-redundant method and with a constant scaling factor. Thereby, the online computation problem [1], as well as the scaling problem are also eliminated.

$$\begin{bmatrix} X_N \\ Y_N \\ Z_N \end{bmatrix} = \begin{bmatrix} K_{dr}\sqrt{X_0^2 + Y_0^2} \\ 0 \\ \arctan\dfrac{Y_0}{X_0} \end{bmatrix} \quad (1)$$

Algorithm 2 DRCORDIC-VM

Input: : $X_{in}, Y_{in}, N, LTAN, K_{dr}^{-1}$
Output: : X_{out}, Z_{out}
X_0=X_{in}, Y_0=Y_{in}, Z_{in}=0, δ_0=1 {Initialization}
if $Y_0 \geq 0$ **then**
  δ_0=1;
else
  δ_0=-1;
end if
for $i = 1 to N$ **do**
  [X, Y, Z]=MICRO-DRCORDIC-VM($X_o, Y_o, Z_o, TLUT(i), \delta_i$)
  if $Y \geq 0$ **then**
    δ_{i+1}=1;
  else
    δ_{i+1}=-1;
  end if
  $X_0 = X$
  $Y_0 = Y$
  $Z_0 = Z$
end for
$X_{out} = X \cdot K_{dr}^{-1}$; Z_{out}=Z

The constant scaling factor for the double-rotation CORDIC is given by $K_{dr} = \prod_{i=1}^{N}(1 + 2^{-2i-2})$ which approximately equals 1.084727 ($K_{dr}^{-1} = 0.9219$), and the maximum and minimum rotation angle $\sum_{i=1}^{N} 2tan^{-1}(2^{-i-1})$ is in the range of [-0.9885,+0.9885].

Alg. 3 shows the computation of the beam phase and magnitude based on online measurements of the gain voltages GV_i and beam position BP_i signals [2], which is illustrated in Fig. 1. The beam phase difference signal ($\Delta Phase$) is obtained from the difference between the phase detection signals P_A and P_B of the gap voltages GV_i and beam positions BP_i signals, respectively.

Algorithm 3 BEAM Phase and Magnitude

Input: : GV_1,GV_2,GV_3,GV_4,BP_1,BP_2,BP_3,BP_4
Output: : $M, \Delta\Phi$
$Q_{1A} = GV_1$; $I_{1A} = GV_2$; $Q_{2A} = GV_3$; $I_{2A} = GV_4$
$Q_{1B} = BP_1$; $I_{1B} = BP_2$; $Q_{2B} = BP_3$; $I_{2B} = BP_4$
$\Delta X_A = Q_{1A} - Q_{2A}$
$\Delta Y_A = I_{1A} - I_{2A}$
[M_A, P_A]=DRCORDIC-VM($\Delta X_A, \Delta Y_A, N, TLUT, K_{dr}^{-1}$)
$\Delta X_B = Q_{1B} - Q_{2B}$
$\Delta Y_B = I_{1B} - I_{2B}$
[M_B, P_B]=DRCORDIC-VM($\Delta X_B, \Delta Y_B, N, TLUT, K_{dr}^{-1}$)
$M = M_B$
$\Delta\Phi = P_A - P_B$

The architecture of the 3-stage double-rotation CORDIC algorithm for the beam phase-magnitude detection is illustrated in Fig. 2. The 3-stage pipeline architecture is designed to improve the performance of the beam-phase detector.

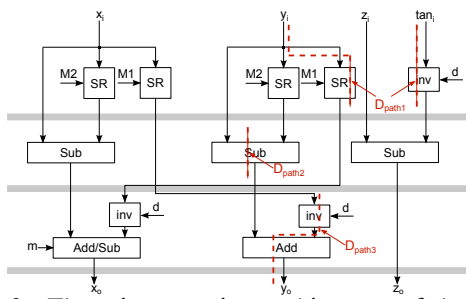


Figure 2: The micro-rotation architecture of the 3-stage double-rotation CORDIC algorithm, where $M_1 = -i$, $M_2 = -2i - 2$ and $d = \delta_i$.

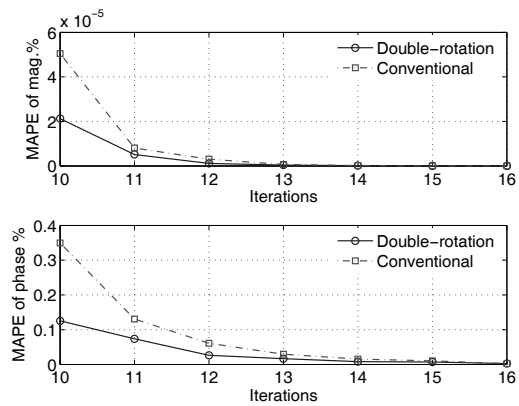


Figure 3: MAPE comparison of the conventional CORDIC and the double-rotation CORDIC.

Figure 3 depicts the performance comparison between the conventional and the double-rotation CORDIC algorithm by using the MAPE metric. MAPE stands for

Mean Absolute Percentage Error and is a statistical measurement of the CORIDC output accuracy formulated by $\frac{1}{n}\sum_{s=1}^{n}|\frac{c_s - m_s}{c_s}|$, where n is the number of measurements, c_s is the CORDIC output and m_s is the ideal output. The figure shows that the double-rotation technique gives better accuracy with the same number of iterations. In other words, for the same MAPE value, the double-rotation CORDIC requires less interations.

Simulation and Synthesis Results

The simulation result of the double-rotation CORDIC algorithm for the beam phase signal detection is depicted in Fig. 4. The figure presents the beam input signal, the reference signal and the the output of the phase detector. The output of the beam phase detector (beam phase signal) is interfered with noisy signal. Therefore, the noise signal should be filtered to acquire the noise-free beam phase signal. The filtering issue is described in the next section.

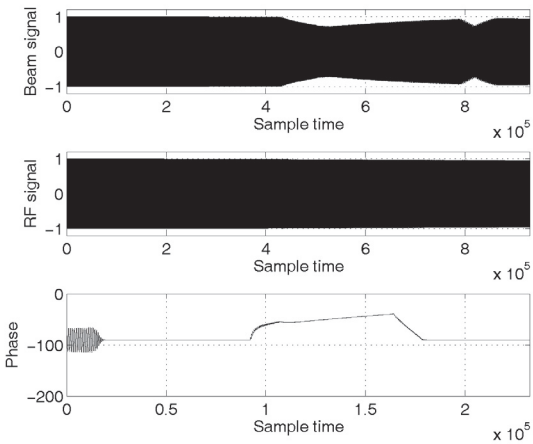


Figure 4: Beam input, reference and beam phase signals.

Table 1 shows the synthesis results of the micro-rotation of the double rotation CORDIC algorithm onto a Xilinx FPGA vlx110t-2ff1738 for different implementations (in terms of the number of pipeline stages). Logic area comparisons between the conventional (CV) and the double-rotaion (DR) CORDIC architectures are presented in the table.

Table 1: Synthesis Results on Xilinx FPGA vlx110t

CORDIC	Utilization			Min. Delay (ns)	Max. Freq. (MHz)
	SReg	SLUT	LUT-FF		
1-state CV	252	0	0	4.68	213.668
2-state CV	258	343	231	3.05	327.836
3-state CV	273	355	239	2.94	340.513
1-state DR	663	0	0	7.713	129.653
2-state DR	258	698	230	5.045	198.210
3-state DR	746	1228	405	4.107	243.496

SReg=Slice Register, SLUT=Slice LUT

PHASE DIFFERENCE SIGNAL FILTERING

Filter Architecture

The noise in the phase difference of the beam signal shown in Alg. 3 is filtered by using adaptive FIR filter in which the filter parameters are adaptively tuned by using a least-mean-square (LMS) algorithm. The hardware architecture of the adaptive FIR filter is designed by using the combination of serial and parallel structures. This combination is made to fulfill the time constraint of the application and the area constraint of the FPGA device. Figure 5 presents the serial-parallel architecture of the adaptive FIR filter with 64 taps. Two registers are controlled by a control unit and are used to store the sampled input signal $x(k-j)$ and the filter parameters w_j. The filter output computation is implemented in serial, while the filter parameter adaptation algorithm is implemented in parallel.

Floating-point adder, multiplier and multiplier-accumulator units were used in the filter architecture. The operands are IEEE single-precision binary floating-point numbers [12], i. e. they have 1 sign bit, 8 exponent bits and 23 mantissa bits. The floating-point adder architecture consists of three stages: exponent difference and alignment, adding/subtracting combined with leading-on-detection and normalization stage.

The floating-point multiplier architecture also consists of three stages: exponent adding and mantissa multiplication, mantissa shifting and normalization stages. In the normalization stages, the arithmetic computing results are normalized into the standard 32-bit (single precision) binary floating point format. The normalization stages include also the zero and infinity detections of the data values according to the IEEE standard [12]. This approach is the same as the one used in other projects documented in available literature [8].

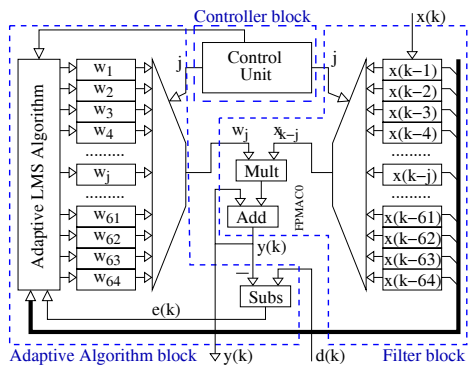


Figure 5: Adaptive FIR Filter architecture with 64 taps.

Simulation and Synthesis Results

Figure 6 shows the input-output signals of the adaptive filter, the desired output signal and the error signal between the filter output and the desired output signals. The main challenge of the adaptive FIR filter hardware in practice

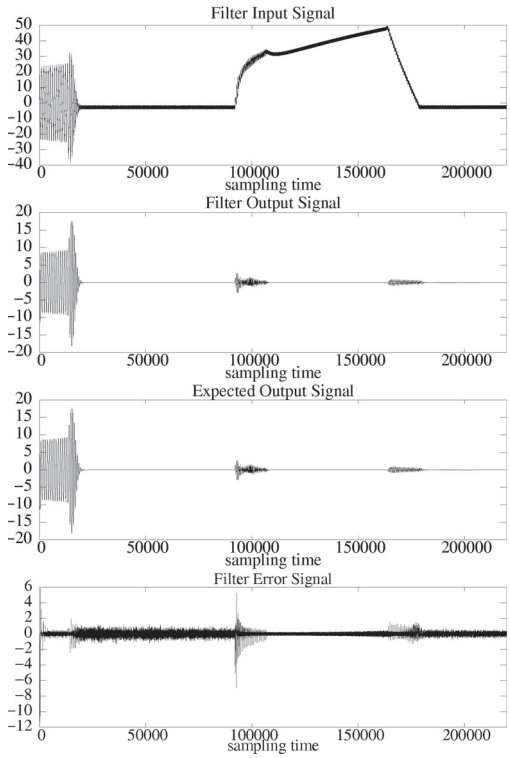


Figure 6: Filter input-output signals.

Table 2: Synthesis Result on FPGA Virtex 5 Device

	Utilization	% of Total
Number of slice registers	38,549	31%
Number of slice LUTs	75,485	61%
Minimum Delay	7.462 ns	
Maximum Frequency	134.018 MHz)	

is the generation of the expected filter output. At present, the expected signal is calculated off-line. In the simulation results shown above, the signal was captured from a system model [9, 10]. We are working on a hardware implementation of a simplified system model that can serve as a reference in the future.

The adaptive FIR filter with LMS algorithm has been implemented on the Virtex5 FPGA device (device type 5vfx200tff1738-2). The device has a total number of 200k logic gates. The synthesis result is shown in Table 2.

CONCLUSIONS AND OUTLOOKS

The beam phase-magnitude hardware detector based on the double-rotation CORDIC algorithm has been presented. Compared to the conventional CORDIC algorithm, the double-rotation CORDIC algorithm provides better accuracy. The adaptive FIR filter architecture that combine serial and parallel computation modes is also presented. The adaptive LMS algorithm is driven by the error signal between the filter output and the desired signal. For now, the desired signal is calculated off-line. In the future, we plan to include a simplified system model in order to supply the adaptive FIR filter with an expected signal that is not known a priori.

ACKNOWLEDGEMENTS

The author would like to thank German Federal Ministry of Education and Research (*Bundesministerium für Bildung and Forschung*) for supporting the work in the framework of Project FAIR (Facility for Antiproton and Ion Research), Grant Number 06DA9028I.

REFERENCES

[1] M. D. Ercegovac, T. Lang "Redundant and on-line CORDIC: Application to matrix triangularization and SVD". *IEEE Transactions on Computers*, pp. 725–740, 1990.

[2] A. Guntoro, P. Zipf, O. Soffke, H. Klingbeil, M. Kumm, and M. Glesner, "Implementation of Realtime and High-speed Phase Detector on FPGA," in *Proc. Int'l Workshop on Applied Reconfigurable Computing*, 2006, pp. 1–11.

[3] H. Klingbeil. "A Fast DSP-Based Phase-Detector for Closed-Loop RF Control in Synchrotrons". *IEEE Trans. Instrumentation and Measurement*, 54(3):1209–1213, June 2005.

[4] H. Klingbeil, D. Lens, M. Mehler, B. Zipfel. "Modeling Longitudinal Oscillations of Bunched Beams in Synchrotrons". http://www.arxiv.org/abs/1011.3957.

[5] H. Klingbeil, B. Zipfel, M. Kumm, and P. Moritz. "A Digital Beam-Phase Control System for Heavy-Ion Synchrotrons". *IEEE Trans. Nuclear Science*, 54(6):2604–2610, Dec. 2007.

[6] D. Lens, H. Klingbeil, T. Gußner, A. Popescu, K. Groß. "Damping of Longitudinal Modes in Heavy-Ion Synchrotrons by RF-Feedback, " in *Proc. IEEE Conf. on Control Applications*, 2010, pp. 1737–1742.

[7] P.K. Meher, J. Valls, Juang Tso-Bing, K. Sridharan, K. Maharatna "50 Years of CORDIC: Algorithms, Architectures, and Applications". *IEEE Transactions, Circuits and Systems*, pp. 1893–1907, 2009.

[8] B. W. Robinson, D. Hernandez-Garduno, and M. Saquib. "Fixed and Floating-Point Implementations of Linear Adaptive Techniques for Predicting Physiological Hand Tremor in Microsurgery". *IEEE Journal of Selected Topics in Signal Processing*, 4(3):659–667, July 2010.

[9] C. Spies, P. Zipf, M. Glesner, and H. Klingbeil, "Bandwidth Requirement Determination for a Digitally Controlled Cavity Synchronisation in a Heavy Ion Synchrotron Using Ptolemy II," in *Proc. Int'l Workshop on Rapid System Prototyping*, 2008, pp. 196–202.

[10] C. Spies, "Model-Based Feasibility Analysis of Digital Beam Phase Control in a Heavy-Ion Synchrotron," in *Proc. Design Automation and Test in Europe, PhD Forum*, 2011.

[11] P. Surapong, M. Glesner, and H. Klingbeil, "Implementation of realtime pipeline-folding 64-tap filter on FPGA," in *PhD-Forum: PhD Research in Microelectronics and Electronics (PRIME)*, 2010, pp. 1–4.

[12] Standards Committee of the IEEE Computer Society. "IEEE Standard for Binary Floating-Point Arithmetic". ANSI/IEEE Std. 754-1985.

[13] N. Takagi, T. Asada, S. Yajima "Redundant CORDIC methods with a constant scale factor for sine and cosine computation". *IEEE Transactions on Computers* pp. 989–995, 1991.

SPI BOARDS PACKAGE, A NEW SET OF ELECTRONIC BOARDS AT SYNCHROTRON SOLEIL

YM. Abiven*, P. Betinelli-Deck, F. Blache, J. Bisou, F. Briquez, A. Chattou, J. Coquet, P. Gourhant, N. Leclercq, P. Monteiro, G. Renaud, JP. Ricaud, L. Roussier,
Synchrotron Soleil, Paris, France

Abstract

SOLEIL is a third generation Synchrotron radiation source located in France near Paris. The Storage Ring currently delivers photon beam to 23 beamlines.

As the machine and beamlines improve their performance over time, new requirements are identified. On the machine side, new implementation for feedforward of electromagnetic undulators is required to improve beam stability [1]. On the beamlines side, a solution is required to synchronize data acquisition with motor position during continuous scan.

In order to provide a simple and modular solution for these applications requiring synchronization, the electronics group has developed a set of electronic boards known as the "SPI board package".

This paper describes the development conducted and the results obtained with this solution.

REQUIREMENTS

Since accelerators and beamlines work and received users, operational feedback has enabled us to identify requirements to improve systems. The two main requirements are the improvement of beam stability during undulator movement and the enhancement of the beamlines scan. The translation of these two requirements in terms of control is as follows:

- Upgrading control of Electromagnetic undulators to improve the feedforward
- Developing an optimized control for EMPHU (rapid undulator combining coils and permanent magnets).
- Processing encoder signals:
  - Interfacing incremental encoder signals with our standard counter board
  - Protocol Conversion
    - incremental encoder signals into « PULSE » and « DIR »
    - single turn absolute encoder to multiturn absolute encoder…
    - generating trigger or synchronization from encoder signals
    - calculation on multiple encoders
    - duplication of encoder signals

Moreover our strategy for these developments not only considers technical aspects but also budget and working load. The R&D activity of the ECA (Electronic Control and Acquisition) group is shared with accelerators and beamline operation. Considering these aspects, it is crucial for a small team with a small budget like ours to coordinate developments well and choose a technical solution that is easy to implement and maintain.

In order to offer a solution to these requirements, the ECA group has implemented the following architecture.

ARCHITECTURE

In the SPI board package architecture (Figure 1), the boards can be connected together in a daisy chain and communicate with the controller via a SPI (Serial Peripheral Interface) Bus. Communication with the control system is done via Ethernet.

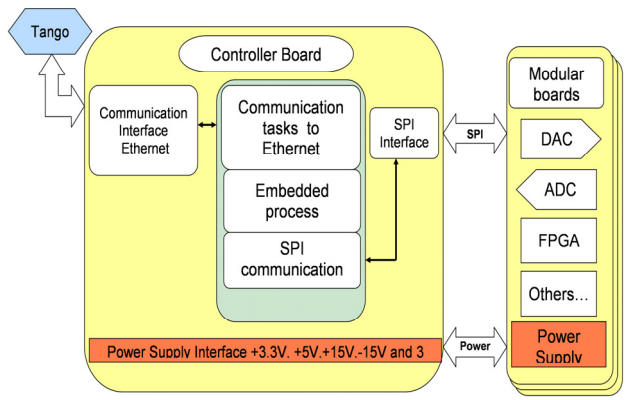


Figure 1: Functional diagram of the SPI BOARD PACKAGE.

The goal of this architecture is to give us a platform on which we can develop specific solutions with simple, reliable and durable tools. The platform is suitable in particular for applications with synchronization requirements, which are managed by implementing the process at low level.

Furthermore, this platform is modular. Each board can be connected with others as needed. And it enables us to deliver solutions for applications with an analog interface or motion interface (encoder side). Finally it's easy to connect it to the Soleil control network.

*yves-marie.abiven@synchrotron-soleil.fr

TECHNICAL CHOICES FOR PROGRAMMABLE COMPONENTS

In order to give a solution for the requirements identified in 2007, initial versions of boards were developed with basic components which allowed us to start quickly and validate the architecture. The first version of the controller board is built around an 8051 microcontroller communicating via Profibus to the supervision. For the board processing encoder signals, we use a basic SPARTRAN III FPGA Xilinx [2].

Microcontroller

Thanks to this experience with the first boards, an upgrade of the controller board called SPICONTROLLER has been approved for a more powerful microcontroller with an Ethernet interface. After analysing the market states for microcontrollers and microprocessors, ARM cortex M3 technology was selected.

The choice of this device takes into account the state of the technology, the price and the investment required for the development tools and test bench.

The device selected for the development is the Texas Instruments [3] LM3S9B96 for the following characteristics: Multipurpose microcontroller used in industry and medical applications: Integrated Ethernet controller, Ethernet 10/100 MAC&PHY, On-chip Memory: 256KB Flash, 96KB SRAM, 80MHz Cortex Processor Core, 2 SPI interfaces.

For the embedded software of the SPICONTROLLER, after evaluating different operating systems and TCP/IP stacks, we selected RTX and TCPnet.

The choice to use this OS and TCP/IP stack is motivated, after evaluation of other OSs and stacks, by the full set of configurations and the ease of configuration via a wizard in the Keil [1] framework.

FPGA (Field Programmable Gate Array)

The FPGA selected for the SPIETBOX was chosen because it had sufficient capacity in terms of pins, cells etc., to implement the first requirement of protocol conversion for encoders (figure 2).

Furthermore this FPGA was already used by the team on the TIMBEL cPCI board [4]. The experience with this component allowed us to quickly design the SPIETBOX. This board is now also used to implement more complex applications such as “Synchronization of Goniometer and Pilatus detector for continuous scan” This application is presented below in the results section.

For that reason, we are envisaging, for future requirements, improving the performances of the FPGA and the input and output of the board.

But before considering an upgrade of the SPIETBOX, a review of the first experience and an analysis of the new requirements must be carried out.

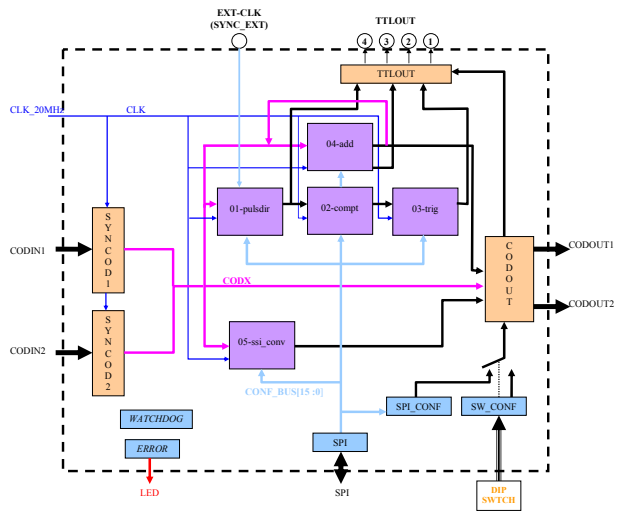


Figure 2: FPGA architecture of protocol conversion for encoders.

BOARDS

To date, the ECA group has developed the following set of boards:

- SPICONTROLLER: controller board with Cortex M3 microcontroller.
- SPIDAC: DAC 4 channels board, 16 bits, ±10V range
- SPIADC: ADC 4 channels board, 16 bits, ±10V range
- SPIETBOX: Processing encoder signal board based on Xilinx FPGA SPARTAN III, 4 Encoder inputs/outputs, 4 TTL Outputs, 1 SPI interface (Works in standalone or connected to SPICONTROLLER

This set of boards is integrated in a low cost standard 3U, 19” crate as shown on Figure 3.

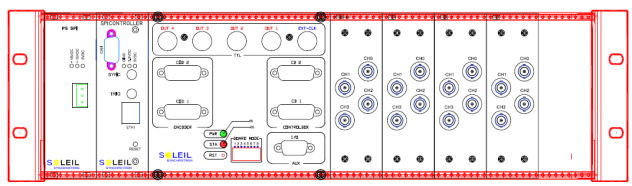


Figure 3**:** SPI Boards integration**.**

SOFTWARE STANDARDIZATION

To ease the integration of the SPICONTROLLER in the control systems, a generic memory structure has been specified. The memory is organized in 32 bits data block Registers with the following organization: State, Status, Error, Command, RW and Read Attributes.

You can have as many blocks as you have identified objects or functions.

This organization of the memory fits with the requirements of the generic Tango [5] device server known as SPIdataviewer. This server is designed to be

easily configured to access the memory of the SPICONTROLLER.

RESULTS

In this part, we describe two applications of the SPI board Package already in production.

Upgrade of the Control of HU256 Electromagnetic Undulator

Each HU256 undulator[6] is driven by a total of fifteen power supplies including three main power supplies, four correction power supplies and eight so-called modulation power supplies. Whereas the main power supplies generate the 256 mm period, 3 m long periodic magnetic field, which makes the electron beam radiate, the correction power supplies create short extremity magnetic fields in order to cancel the residual closed orbit distortion caused by the undulator magnetic defects. Finally the modulation power supplies modulate some field peaks, allowing the so-called quasi-periodic mode, in order to decrease the intensity of the radiation higher harmonics.

Since they were installed, the three HU256 undulators used to be driven using the Profibus interface, but many transitory variations could be observed on the electron beam orbit, perturbing the radiation stability for all the beamlines as shown on Figure 3. An analysis determined that these effects were caused by non-synchronization between the different power supplies.

The upgrade of the system entails generating synchronized analog signals with the SPICONTROLLER in order to drive the power supplies. This evolution allowed improved synchronization of the power supplies, leading to a decreasing of the orbit transitory perturbation and thus as shown on Figure 4, reduced use of the SOLEIL storage ring fast orbit feedback.

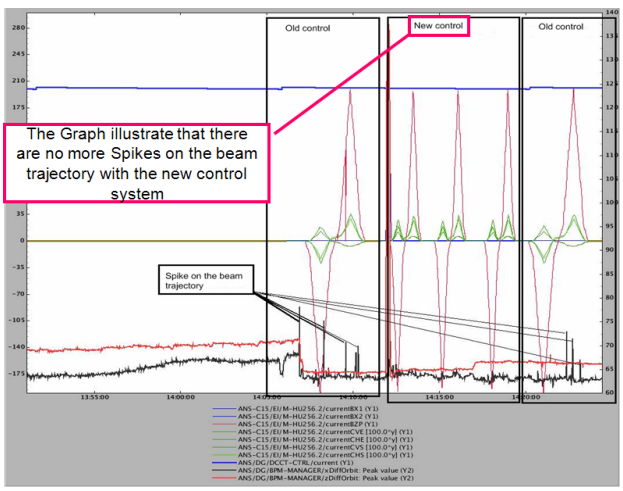


Figure 4: Acquisition showing the difference between old and new control system of HU256 undulators.

Synchronization of Goniometer and Pilatus Detector for Continuous Scan

On PX1 beamline [7], when a continuous scan is done acquiring Frames on Pilatus detector during goniometer movement, the following architecture (Figure 5) has been implemented to:

- Synchronize goniometer position, Shutter aperture and Pilatus acquisition
- Check that acquisition is well done.

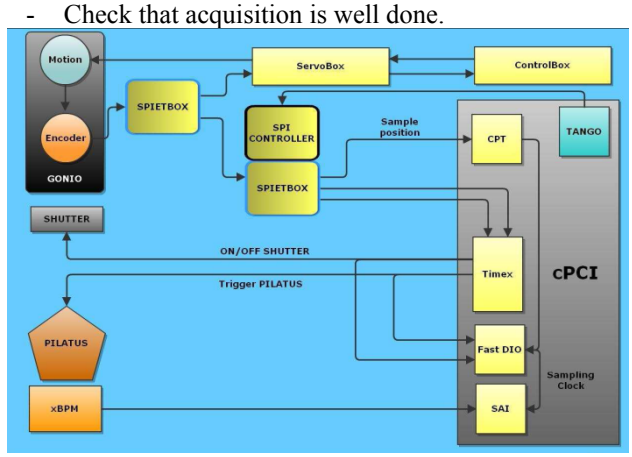


Figure 5: Architecture for PX1acquisition with Pilatus.

The Pilatus detector allows four acquisition modes that are four different ways to synchronize acquisition. In this architecture these synchronizations are implemented in the SPIETBOX managing the following signals: encoder position, trigger to and from the Pilatus, shutter trigger.

The SPICONTROLLER is used to configure the different modes and to transfer some information along the process. The information coming from the SPIETBOX and beam intensity from XBPM are logged for diagnosis along the process.

At the moment in the most used acquisition mode with Pilatus internal clock (Figure 6), the SPIETBOX sends a trigger to start the detector which uses its own internal clock to take the images.

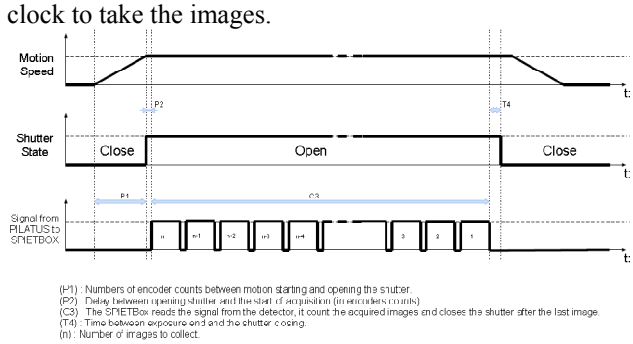


Figure 6: Timing of the most use triggering mode.

The advantages of this architecture using the SPIETBOX are to have good correlation between data acquisition and motor movement thanks to hardware synchronization. It enables the scientists to have a quick validation of the acquisition, thanks to the logging of all experimental conditions. Lastly, it simplifies control of the Pilatus detector's modes.

CONCLUSION

This platform allows us to embed process close to the hardware with open tools. Thanks to this solution we improve the performances of applications requiring beam stability or synchronization of multiple detectors acquisition.

The next step to increase calculation possibilities with this platform is to upgrade the SPIETBOX but we are also considering developing a co-processing board for SPICONTROLLER based on DSP or FPGA.

Future applications using this solution will include:

- Upgrade of HU640 Undulators to improve switching performance.
- Control of Emphu Undulators.

Finally all these boards are interesting for other synchrotron facilities that have the same requirements. In order to collaborate with them, the ECA group shares a part of these developments through the “Open Hardware Repository (OHR) [8]” platform.

REFERENCES

[1] N.Hubert & al, SOLEIL Beam Orbit Stability Improvements, Proceedings of IPAC11, TUPC068, San Sebastian, SPAIN
[2] Xilinx Manufacturer, www.xilinx.com
[3] Texas Instruments, www.ti.com
[4] JP.RICAUD &Al, The TimBel Synchronization Board for Time Resolved Experiments at Synchrotron SOLEIL Proceeding of ICALEPS 2011, WEPMS026, Grenoble, France
[5] The TANGO framework, http://www.tango-controls.org
[6] F. Briquez & al, COMMISSIONING OF THE ELECTROMAGNETIC INSERTION DEVICES AT SOLEIL. Proceedings of EPAC08, Genoa, Italy, p2237-2239
[7] www.synchrotron-soleil.fr/Recherche/LignesLumiere/PROXIMA1
[8] www.ohwr.org

FABRIC MANAGEMENT WITH DISKLESS SERVERS AND QUATTOR IN LHCb

P. Schweitzer*, E. Bonaccorsi, L. Brarda, N. Neufeld, CERN, Geneva, Switzerland.

Abstract

At CERN LHCb experiment (Large Hadron Collider beauty), in order to reconstitute and filter events, a huge computing facility is required (currently ~1500 nodes). This computing farm, running SLC (Scientific Linux CERN) [1], a Red Hat Enterprise Linux (RHEL) [2] derivate, is net booted and managed using Quattor [3] to allow easy maintenance. LHCb is also using around 500 diskless Credit Card PCs (CCPCs) embedded in the front end electronic cards. To go farther than the limited Red Hat provided (and now deprecated) netboot tools, a new solution has been designed and will be shortly presented in this paper.

LINUX DISKLESS

All these computing nodes get their operating system (Linux) from some dedicated servers (each server is controlling a slice of the farm).

Boot Process

The node's boot ROM sends a DHCP query. Its server replies with the IP address to use and the location of the pxelinux.0 stage 2 loader file. The node gets this file by TFTP and starts it. The stage 2 boot loader then loads the Linux kernel and initial ramdisk and gives the control to the initialisation script located in the ramdisk.

This script has to setup the networking, mount the root file system from the server and switch to this NFS root. It then passes the control to the usual Linux initialisation scripts.

For many reasons, the running Linux needs to write files in many locations. We will now see the Red Hat way of making these files writeable and the problems and limitations of this method. Then we will talk about what was developed to circumvent these problems and limitations.

Diskless: Red Hat Way

Up to RHEL5, Red Hat had a package named system-config-netboot to setup diskless servers. It was a set of python and bash scripts that were setting up the dhcpd and tftpd servers, customising the shared root file system and making the initial ramdisk for the diskless nodes.

To make some files of the root file system writeable for the nodes, with possibly different contents, the initialisation script from this package was making the following actions after having mounted the root file system:

- Mount the 'snapshot' directory from the server, in read/write mode. This directory contains one sub-directory per node and two files with the list of the files that need to be writable.
- Remount (using the bind mount option) each of these files from the node's snapshot to the root file system.

There are two problems with this method:

- Only files or folders of the fixed list can be writeable. To add a file to that list, we have to reboot the nodes after the file list modification.
- The mount table is 'polluted' by all these remounts.

Diskless: New LHCb Way

To add flexibility to the diskless nodes handling, we had the idea of using file system union, which is usually used on Linux live media (CDs, DVDs or USB keys). As LHCb is using Scientific Linux CERN 5 in production and SLC 6 has been released and will go in production, the new system had to support both.

FILE SYSTEMS UNION

The principle of the file system union is to join several file systems, at least one read-only (generally) and one read-write and use that union as a normal Linux file system. The behaviour is the following:

- When creating a file, the file is directly created on the read-write file system.
- When writing a file, if the file only exists only on the read-only file system: the modified file (called copyup) is written on the read-write file system with the same name and hides the read-only version.
- When erasing a file, if the file only exists on the read-only file system: a whiteout file (filename prefixed by '.wh.') is written on the read-write file system to hide the file.

Table 1: Union File Systems Evaluation

		Feature				
		mode	**Copy on write**	**NFS branches**	**Kernel specific rebuild**	**Metadata separation**
Implementation	UnionFS [4]	kernel	Yes	Yes	Yes	No
	Aufs [5]	kernel	Yes	No	Yes	No
	Funionfs [6]	FUSE [8]	Should be	Yes	No	No
	Unionfs-fuse [7]	FUSE	Yes	Yes	No	No

- Renaming a file is done by copying the old file under the new name on the read-write file system and hiding the old file with a whiteout file.

Several implementations of file system fusion were evaluated. Table 1 shows the result of this evaluation.

We first tried to make the union on the nodes during diskless initialisation but finally choose to do it on the server, and NFS exports the "unioned" file system. Client side union was just using too much memory on the CCPCs.

An Union File System Designed for Diskless

While evaluating union file systems implementations, it became clear that none was perfect for net booted nodes. All were designed with totally different goals than ours.

One of the big problems was that too many copyups were made on the read-write file system.

So we decided to implement an union file system designed for diskless systems, with the following functionalities:

- union between only one read-only and one read-write file systems
- if only the file metadata are modified, then do not copy the whole file on the read-write files system but only the metadata (stored with a file named as the file itself prefixed by '.me.')
- check when files on the read-write file system can be removed

As a proof of concept, a first version of this new union file system, named PierreFS (until we find a better name) as been developed and successfully tested: the read-write file system was only containing configuration files configured by our Quattor configuration system and runtime files in /var. Even library pre-linking, which was making a lot of copyups in other implementations did none in PierreFS.

The next step will be the implementation of this file system directly as a module for the kernel, without the use of FUSE. This is a huge work that will make it more efficient using less memory. Of course, it will have to be compatible for both SLC5 and SLC6 kernels.

QUATTOR COMPONENT

The LHCb experiment is using the Quattor toolkit to manage the configuration of all its Linux servers and nodes.

In the previous Quattor component in charge of the diskless configuration, supporting the Red Hat way, many things were done by the Red Hat system-config-netboot package. Some of the scripts in this package were buggy, with no output in case of problems. Red Hat totally removed that package in RHEL6, the distribution SLC6 is based on.

As the Quattor component itself was not really maintainable, it was decided to completely rewrite it to make it more efficient and modern, and handle completely the new diskless model, of course without using the system-config-netboot scripts.

System-config-netboot Replacement

In this package, several scripts were used by the Quattor component:

- pxeos & updateDiskless were in charge of copying the kernel and building the diskless specific initial ramdisk (initrd.img): in the new component, updateDiskless repackaged, is still used for SLC5 while rracut is used for SLC6. CCPCs are a special case as they do not support pxe: the wraplinux utility is used to create the nbi file containing the kernel and the ramdisk together.
- pxeboot was in charge of making the pxe configuration files for the nodes. The functionality is now in the Quattor component.
- mkdiskless was used to customise the shared root file system for diskless use. The functionality is now in the Quattor component, with a possible call to an external script.

Other Improvements

Rewriting the component gave us the possibility to greatly improve it and simplify the setup of diskless servers.

In the previous setup, the root file system shared by all nodes was created by copying the server root file system

at the end of its installation. It was not possible to have clients with a different architecture or a different version of the Linux distribution. We added the functionality to create the shared root file system for any Linux distribution and architecture, by using the '--installroot' parameter of the yum command. The only thing we have to do is to provide the component a yum configuration file. We can even have one server with many shared root file systems, each with different versions of the distribution and/or different settings. Like this, we can have a node running SLC5, just change some settings on the server, and then have a restart under SLC6.

The dhcpd configuration part now has better support for subnets and the generated files follows more closely the standard, using dhcpd groups to avoid repeating the same options for all hosts.

The diskless component also keeps track of the created and administrated nodes. That way, in case of deletion of a node in the configuration, it can properly remove it, without leaving leftovers. It also handles configuration changes on the nodes (MAC address change, IP address change).

Changes to Another Component

To configure the shared root file systems, the Quattor component runs the Quattor utilities "chrooted" in these root file systems. To ensure proper run of Quattor inside of Quattor, the proc pseudo file system is mounted just before the change root and the Quattor run. This allows locks consistency, but also enables mounting/dismounting. That is why the Quattor NFS component has been fixed to add an option that makes this component only write configuration and not attempt any mount or dismount.

CCPC Support

This new diskless component also comes with a great improvement since it adds CCPCs management into Quattor. Some more tweaks are done on the CCPCs shared root file system and snapshots, as disabling Quattor on them (that would have taken too many resources).

SIDE EFFECTS

On a side note, the use of a FUSE file system as root file system has exposed a bug in the Linux kernel (that appears to be well-known). A workaround has been deployed in the ramdisk init script.

Also, the use of this file system helped revealing security vulnerability in the Linux kernel. Effects of that security vulnerability are really limited and only allow someone who does not have access on a directory to read its contents. Since it is hard to reproduce it out of the box, it has not been reported yet (no easy test-case).

CONCLUSIONS

The concept of using file system union to make diskless systems easier to manage has been successfully tested. We even have new use cases where the way of doing diskless nodes system does not work (eg: Dell management tools, which want to write some inventory files in many places).

The new Quattor diskless component, together with PierreFS will go in production during the next winter shutdown. It will allow us to better manage the Credit Card PCs, which are not supported by the current system. It will also ease the management of the LHCb event filtering farm. With the support of several Linux distributions on the same server, we will be able to easily test SLC6 on the farm nodes, with the ability to go back to SLC5 with a simple reboot of the nodes.

The new file system has been published on SourceForge [9] and the work to integrate it to the kernel will start in October 2011.

REFERENCES

[1] http://cern.ch/linux
[2] http://www.redhat.com/rhel/
[3] http://www.quattor.org
[4] http://www.fsl.cs.sunysb.edu/project-unionfs.html
[5] http://aufs.sourceforge.net/
[6] http://funionfs.apiou.org/
[7] http://podgorny.cz/moin/UnionFsFuse
[8] http://fuse.sourceforge.net/
[9] http://sourceforge.net/projects/pierrefs/

MANAGEMENT TOOLS FOR DISTRIBUTED CONTROL SYSTEM IN KSTAR

Sangil Lee, Jinseop Park, Jaesic Hong, Mikyung Park and Sangwon Yun
National Fusion Research Institute, Gwahangno 113, Yuseong-Gu, Daejeon 305-333, KOREA

Abstract

The integrated control system of the Korea Superconducting Tokamak Advanced Research (KSTAR) has been developed with distributed control systems based on Experimental Physics and Industrial Control System (EPICS) middleware. It has the essential role of remote operation, supervising of tokamak device and conducting of plasma experiments without any interruption. Therefore, the availability of the control system directly impacts on the entire device performance. For the non-interrupted operation of the KSTAR control system, we have developed a tool named as Control System Monitoring (CSM) to monitor the resources of EPICS Input/Output Controller (IOC) servers (utilization of memory, cpu, disk, network, user-defined process and system-defined process), the soundness of storage systems (storage utilization, storage status), the status of network switches using Simple Network Management Protocol (SNMP), the network connection status of every local control sever using Internet Control Message Protocol (ICMP), and the operation environment of the main control room and the computer room (temperature, humidity, electricity) in real time. When abnormal conditions or faults are detected by the CSM, it alerts abnormal or fault alarms to operators. Especially, if critical fault related to the data storage occurs, the CSM sends the simple messages to operator's mobile phone. In addition to the CSM, other tools, which are subversion for software version control and vmware for the virtualized IT infrastructure, will be introduced.

INTRODUCTION

EPICS [1] was adopted as a middleware of the KSTAR Integrated Control System (KICS) [2,3] to integrate and control a lot of network-based distributed systems for tokamak operation and plasma experiment. As well known, EPICS is a set of open source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific facilities such as particle accelerators, telescopes and other large scientific experiments. Also, Qt-SDK [4] was used as a tool for developing user interface in KSTAR. The KSTAR Widget Toolkit (KWT) [5,6], which is a name of libraries developed on Qt-SDK, had been considered in the view of the following functions:

- Easiness of development
- Maximum effectiveness for user interface
- Performance of visibility
- Software reusability
- Flexibility for requirements of operators
- Independence from OS Platform

CONTROL SYSTEM MONITORING

The KSTAR control system is composed of a lot of local control systems distributed in the vicinity of the KSTAR to operate tokamak, perform experiments and obtain data. The availability of the control system directly results in success or failure of tokamak operation and plasma experiments whereby the Control System Monitoring (CSM) was developed for the purpose of monitoring and supervising the entire control system and IT infrastructure. The CSM consists of two KWT widgets; "BlinkLine" and "CABlinkLabel". The two widgets among many KSTAR widgets were developed only for use in the CSM. Figure 1 shows the configuration of software module configuration for the CSM.

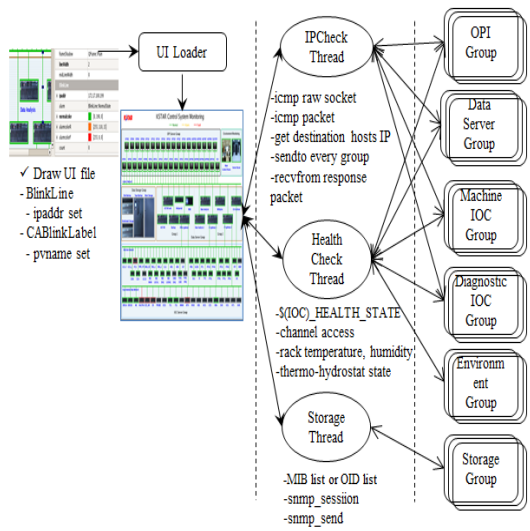


Figure 1: Control System Monitoring software module.

Network Monitoring

"BlinkLine" widget has "ipaddr" property to write IP address for monitoring the network status of IOC server. If "BlinkLine" widget flashes in red, it means that the IOC server has some network problems or has gone into shutdown state. In Figure 1, the "IPCheck Thread" gets IP address from "ipaddr" property of all "BlinkLine" widgets and sends ICMP messages to all host destinations using the "sendto" UDP socket at the same time. The ICMP messages are typically generated for responding to errors in IP datagrams, for diagnosis or routing purposes. In addition, ICMP errors are always reported to the source IP address of the originating datagram. After sending ICMP messages, "IPCheck Thread" receives the response messages using "recvfrom" UDP socket. The request IP list and the response IP list are the same in the normal

state of all networks and all "BlinkLine" widgets display in green. If a problem happens in the connections of some IP addresses, the "BlinkLine" widgets related to those IP addresses will blink in red or orange color.

Server Group Monitoring

The "CABlinkLabel" widget has two alarm states. One is the alarm state blinking in yellow for reporting abnormal status in IOC server's resource. The other is the alarm state blinking in red for IOC server problem. Left-clicking on the "CABlinkLabel" widget generates a resource dialog window to inform a user about the resource status of the IOC server in detail. All servers have their own resource information in the form of EPICS record as follows:

- CPU Load (Idle, Nice, System, User)
- Memory (Available, Used, Free, Swap)
- Heartbeat Count
- System Information (IP address, Up time, Disk, etc)
- Network Packet (In-bound, Out-bound)

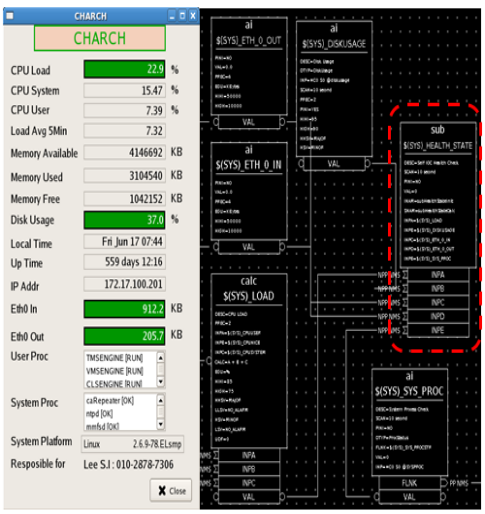


Figure 2: Heath state logic EPICS database.

IOC Health State Logic: $(IOC)_HEALTH_STATE

- Alarm:
  - CPU Load >= 85% or
  - Disk Usage >= 95% or
  - Ethernet Packet >= 20Mbytes or
  - User-defined System Process bit count >= 1
- Normal:
  - CPU Load < 85% and
  - Disk Usage < 95% and
  - Ethernet Packet < 20Mbytes and
  - User-defined System Process bit count == 0

Environment Monitoring

In order to use stable IT resources, it is important to maintain an optimal environment in the control room and the computer room. We have used National Instrument (NI) DAQ modules (cFP-2120, NI cFP-RTD-124 and cFP-DI-304) to monitor the temperature of server racks, the relay signals of thermo-hydrostat and the water leak sensing signals of High Performance Computer (HPC). As shown in Fig. 2, EPICS channel access can be achieved through Ethernet communication between EPIC IOC driver support module and NI module.

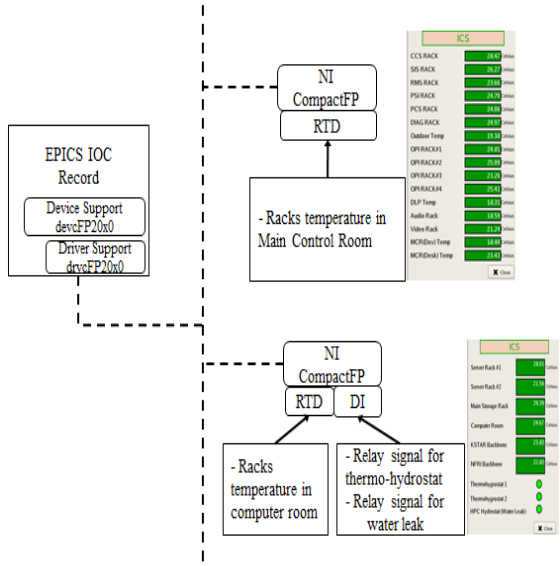


Figure 3: Environment monitoring using NI cFP.

The two "CABlinkLabel" assigned to "Main Control Room" and "Server Room" in Figure 3 blink according to the following alarm logic.

Environment Alarm Logic:

- Main Control Room: ICS_CTRLROOM_TEMP
  - Alarm: 1 rack's temp among 16 racks' temp >= 32 °C
  - Normal: every rack temp < 29 °C
- Computer Room: ICS_SERVERROOM
  - Alarm: 1 rack's temp among 6 racks' temp >= 32 °C or 1 thermo-hydrostat of 2 thermo-hydrostats == alarm or water leak for HPC cooling line == alarm
  - Normal: every rack temp < 29 °C and 1 thermo-hydrostat of 2 thermo-hydrostats == alarm and water leak for HPC cooling line == alarm

Storage System Monitoring

The storage system monitoring has not been fully implemented so far. Some modules were developing using SNMP which is the standard operations and maintenance protocol for the Internet. The SNMP-based management not only produces management solutions for systems, applications, complex devices, and environment control systems, but also provides the Internet management solutions supporting web services. The storage system of KSTAR is also in the form of a management system using SNMP. "Storage Thread" in Figure 1 is being implemented the storage system monitoring module by using the Management Information Base (MIB) that is provided by storage manufacturer. Table 1 shows all IT resources being monitored by the CSM for KSTAR operation.

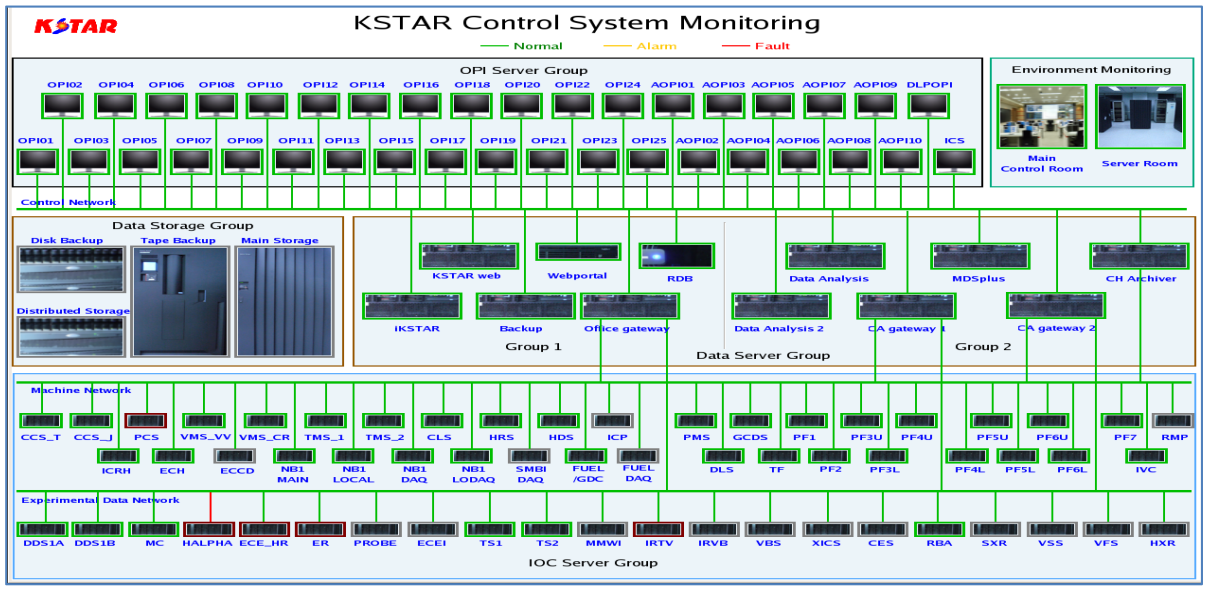


Figure 4: KSTAR Control System Monitoring user interface.

Table 1: IT Resources for KSTAR Operation

Group	Number of server	OS Platform	Network
OPI	35 Servers	Linux	Control
Data	12 Servers	Linux/Window	Control
Plant IOC	37 Servers	Linux/Window	Machine
Diagnostic IOC	21 Servers	Linux/Window	Experiment
Storage	4 Storages		Local
Environment			Control

The Figure 4 shows that the CSM is supervising the entire control system and IT infrastructure for KSTAR operation.

Short Message Service (SMS) System

SMS is a text messaging service component of mobile communication using standardized communication protocols that allow the exchange of shot messages between mobile phone devices (http://en.wikipedia.org/wiki/SMS). Basically the KSTAR Supervisory Interlock System (SIS) manages all the alarms received from the 13 local interlock subsystems and transmits simple alarm messages to operators in charge of the systems which operate for 24 hours a day, such as vacuum pumping system, cryogenic systems, control systems and so on. When the CSM detects any serious problem in main servers like a system shutdown, it notifies the problem to the SIS at once via hard-wired signals. Next, the SIS transmits the alarm messages to a SMS server through RS232 communication and the SMS server relays the alarm messages to CDMA devices. Finally, the operators received the messages correspond to the problem according to emergency procedures.

VERSION CONTROL AND SOFTWARE DEPLOY

Overall, software version control is very important. In particular, it is more important in case that many developers must cooperate with each other or development should be carried out under the distributed environment over a long period of time. The first use case of a tool for software version control was Concurrent Version System (CVS). Later, it was changed to Subversion [7] for some merits as follow:

- Commit unit as work had changed, not a file
- Atomic commits
- Binary file support
- File and directory renames with Unicode.
- Directory version control

Source Code Version Control

Software development in a distributed environment means that the developed codes are distributed more widely and possibility of wrong programs may increase. To create an EPICS IOC, we generate many kinds of files such as c/c++, h, db/vdb, dbd, config, template, st/stt, makefile and so on. It is likely to cause some bugs in programs if we modify or change these different types of files after a long time without software version control. In addition, the necessity of software version control becomes more prominent when program development is carried out by outside contractors.

Software Deploy Management using Subversion

Subversion supports the binary file formats for version control. As shown in Table 1, there are 105 servers for KSTAR operation. Deploying and compiling the source code in every server are very difficult and often a large burden. For these reasons, we installed the linux OS as far as possible to maintain the binary file compatibility and

binary files generated by linux machines work properly in all linux servers except for only a few cases. Therefore, it is possible to deploy the binary modules (libraries or execution files) even if they are generated by a virtualized development server. In particular, it is more suitable for deploying OPI programs because they are modified frequently. The following is the procedure for deploying the binary file.

- Software development in virtualized developing server (source code development and compile)
- Software testing in virtualized distributed servers (software verifying in runtime)
- Commits binary software in SVN server
- Deploying binary software (object files or execution files)

VIRTUALIZATION FOR DISTRIBUTED ENVIRONMENT

Recently, green IT has emerged as a buzzword in the IT industrial field, and among the green IT technologies the computer virtualization technology has the lead. Virtualization of computer (server and desktop) within a limited environment increases the effectiveness of IT resources and reduces costs. As hardware performance and virtualization technology have been improved, a virtual machine has been substituted for a stand-alone server. For the purposes of server redundancy and cost saving, KSTAR has also adopted the virtualization in servers being less intensive against CPU utilization.

The virtualized servers are implemented using ESX/ESXi hypervisor of vmware [8].

Servers virtualized in KSTAR:

- Virtualized OPI servers for remote experiment participants (10 remote OPI servers)
- Virtualized servers for software testing
- Virtualized development servers
- Operation of virtualized EPICS gateway servers

To use the reliable and verified software for KSTAR, new software should be tested and verified first in the virtualized environments which are simulating the real environments before application. In the future, this virtualization will be extended to the data service servers after testing and pre-verification.

FUTURE WORKS

As the KSTAR experiment campaigns have continued, the longer pulse plasma has been achieved and the more data has been obtained. It has led to lacks in network bandwidth and server resources; moreover the shortages become more prominent in the 2011 campaign. To mitigate the insufficiency and enhance the performance of the infrastructure, we will perform the upgrades as follows:

- Completion implementation of storage monitoring system using SNMP in the CSM
- Consideration of clustering server system
- Implementation of fault tolerant server
- Load balancing for data service
- Virtualization for data service

CONCLUSIONS

The CSM has been developed using several standard protocols for monitoring and supervising the entire control system and IT infrastructure to assist the non-interruptible tokamak operation and successful experiments of KSTAR. The faults or alarms detected by the CSM are immediately reported to the operators by the helps of the supervisory interlock system and the SMS system. The operators then quickly restored the problems according to the emergency procedure. As a result of this process, the KSTAR was able to perform the continuous operation and experiment without interruption during the four months. The use of subversion for software version control minimized the human errors due to the wrong software management and helped the stable operation of KSTAR. Through the virtualization adopted in the KSTAR, the time and cost for consisting of IT infrastructure was reduced for the sudden demands of operation. Overall, software testing into the virtualized environment before applying to the real environment increased the stability of the KSTAR operation.

REFERENCES

[1] EPICS (http://www.aps.anl.gov/epics/)
[2] K.H.Kim, et. Al., The KSTAR Integrated control system based on EPICS, Fusion. Eng. Des. Vol.81 (2006), pp 1829-1833.
[3] Mikyung Park, et. Al. Overview of KSTAR Integrated Control System, Nuclear Engineering and Technology vol.40 (2008), pp 451-458
[4] Qt-SDK (http://qt.nokia.com/)
[1] S.Baek, et.al.., “KSTAR widget toolkit using Qt library for the EPICS-based control system” Proceedings of ICALEPCS (2009).
[6] KWT (http://kwt.sourceforge.net/)
[7] Subversion (http://subversion.apache.org/)
[8] Virtualization (http://www.vmware.com/)

RELIABILITY IN A WHITE RABBIT NETWORK

Maciej Lipiński, Javier Serrano, Tomasz Włostowski, CERN, Geneva, Switzerland
Cesar Prados, GSI, Darmstadt, Germany

Abstract

White Rabbit (WR) is a time-deterministic, low-latency Ethernet-based network which enables transparent, subns accuracy timing distribution. It is being developed to replace the General Machine Timing (GMT) system currently used at CERN and will become the foundation for the control system of the Facility for Antiproton and Ion Research (FAIR) at GSI. High reliability is an important issue in WR's design, since unavailability of the accelerator's control system will directly translate into expensive downtime of the machine. A typical WR network is required to lose not more than a single message per year. Due to WR's complexity, the translation of this real-world-requirement into a reliability-requirement constitutes an interesting issue on its own – a WR network is considered functional only if it provides all its services to all its clients at any time. This paper defines reliability in WR and describes how it was addressed by dividing it into sub-domains: deterministic packet delivery, data resilience, topology redundancy and clock resilience. The studies show that the Mean Time Between Failure (MTBF) of the WR Network is the main factor affecting its reliability. Therefore, probability calculations for different topologies were performed using the "Fault Tree analysis" and analytic estimations. Results of the study show that the requirements of WR are demanding. Design changes might be needed and further in-depth studies required, e.g. Monte Carlo simulations. Therefore, a direction for further investigations is proposed.

INTRODUCTION

The WR project is a multi-laboratory, multi-company, international effort to create a universal fieldbus for control and timing systems to be used at CERN, GSI and possibly other such facilities. The rationale behind WR, the choice of the technologies and technical details of its functioning have been already described in a number of papers [1], [2], [3]. The resilience and robustness is one of the key features of any fieldbus. This article presents a study on the reliability of a White Rabbit Network (WRN) assuming a basic knowledge about WR.

Reliability is defined as the ability of a system to provide its services to clients under both routine and abnormal circumstances. It can be estimated by calculating the probability of the system's failure (P_f). The lesser the probability of WRN failure, the higher its reliability. Thus, in this article we identify critical services of a WRN based on the study of WR's requirements. Then, we analyze each critical service to identify possible reasons for their failure and propose targeted counter-measures to increase reliability. Finally, their impact on the overall system reliability is studied to identify the highest contributor and the focus for the further studies.

DEFINITION OF RELIABILITY IN A WRN

A WRN, consisting of White Rabbit Switches (switches) connected by fiber or copper, is meant to transport information among White Rabbit Nodes (nodes). We distinguish two types of information distributed over the WRN: (1) *Timing* (frequency and Coordinated Universal Time) and (2) *Data* (the Ethernet traffic). This translates into two types of services provided by the WRN which have their own requirements and can be handled separately. The requirements are defined by GSI and CERN as the prospective users of WR to control their accelerators.

Timing Distribution

Timing is distributed in the WRN from a switch/node called Timing Master (TM) to all the other nodes/switches in the network. All the devices in the WRN lock their frequency (syntonize) and adjust their local clocks (synchronize) to that of the TM. The deviation between the clock of the TM and that of any other node/switch is called **accuracy**. A stable and continuous synchronization of all the nodes with an appropriate accuracy is the key requirement of the Timing Distribution in the WRN.

Data Distribution

The critical data distributed over the WRN is the one carrying sets of commands (events) which are organized into Control Messages (CM). The CMs are sent by a privileged node (Data Master, DM) in the payload of the Ethernet frame(s). Therefore, the Data Distribution in the WRN is broken into (1) *Control Data (CD)* – the Ethernet frames carrying CMs, critical, and (2) *Standard Data (SD)* – the Ethernet frames which do not carry CMs, non-critical. The reliability of the WRN depends on the successful delivery of the CD to all the designated nodes. The CMs are always broadcast within a VLAN , which can span the entire network. The worst-case upper bound of their delivery latency from the DM to any node in the network, regardless of it's location (**maximum distance from the DM**), is required to be guaranteed by the network – this is **a determinism** requirement.

Reliability of the WRN

The reliability of the WRN relies on the **deterministic** delivery of the CD to all the designated nodes and their

sufficiently **accurate and stable synchronization**. This means that the WRN is considered non-functional if one or more of the following occur:

- A node is synchronized with insufficient accuracy.
- A designated node receives corrupted CD or no CD.
- The upper-bound delivery latency has been exceeded.

Unreliability is translated into the number of CMs considered lost (not delivered, delivered corrupted or in a non-deterministic way) in a given period of time. During this time, the synchronization must be always of the required quality. Quantitative requirements of the accelerator facilities are listed in Table 1.

Table 1: GSI's and CERN's Requirements Summary

Requirement	GSI	CERN
Max latency	100 µs	1000 µs
CM failure rate	$3.17 * 10^{-12}$	$3.17 * 10^{-11}$
CMs lost per year	1	1
d_{max} from DM	2 km	10 km
CM size	200–500 bytes	1200–5000 bytes
Accuracy	probably 8 ns	1 µs to 2 ns

FAILURE STUDY

One of the main possible reasons for WRN failure, which affects both Timing and Data Distribution, is a malfunction of its elements (switches or links). Since the distribution of information in the WRN is of one-to-all character (Data/Timing Master to all nodes), all the elements of the WRN are considered Single Points of Failure (SPoF)[4]. Malfunction of any SPoF results in failure of the entire system. SPoFs can be eliminated by introducing redundancy of the system components. Due to its special features (distribution of frequency over physical layer) and strict requirements (determinism, low data loss), the number of possible redundant topologies of the WRN is restricted, as explained in the following sections.

Imperfections of the physical medium as well as switching between redundant elements of the network (which takes time) can cause loss or corruption of data. The deterministic and mostly broadcast character of the data distribution in the WRN enforces application of the Forward Error Correction (FEC) – adding redundant information on transmission to enable recovery of lost or corrupted data on reception. This brings constant data overhead and the probability that the added redundancy is not sufficient to recover the data. However, it is the price to pay for ensuring low latency and determinism of data delivery in the WRN.

The delivery latency of an Ethernet frame varies with cable length and the number of hops (switches) it has to traverse to reach its destination, the traffic load on the way and the assigned Class of Service (CoS). Therefore, to ensure the required determinism of the CD delivery, we need to make sure that there is no congestion of Ethernet frames carrying CMs. Moreover, the number of hops (the latency introduced by them) needs to be sufficiently small, which can be done by restricting the topology.

The resilience of the Clock Distribution translates into continuous and stable synchronization of all the nodes and switches in the WRN (Table 1). Although, the network redundancy eliminates SPoFs, the switch-over between redundant elements might introduce instability and render the network unreliable despite the costly redundancy. Therefore, a seamless switch-over between redundant clock paths needs to be ensured. Another reason for the deterioration of the synchronization accuracy is the variation of external conditions (e.g. temperature) which needs to be compensated.

DETERMINISM

A carefully configured and properly used WRN offers deterministic Ethernet frame delivery thanks to the implementation of CoS and the fact that the delay introduced by the switch can be verified by analysis of **publicly available source code** [5]. Such analyses were performed to verify the worst-case upper bound delivery latency of a CM against the requirements listed in the Table 1. The results, presented in Table 2 (*Store-and-forward* column), take into account the fact that a CM is encoded into 4 Ethernet frames (as required by the FEC and described in the next Section), it is sent with the highest priority (CoS) and it always traverses 3 hops.

Table 2: Control Message(CM) Deliver Latency Estimations

CM size	CM deliver latency			
	Store-and-forward		Cut-through	
	GSI	CERN	GSI	CERN
500 bytes	221 µs	283 µs	76 µs	118 µs
1500 bytes	285 µs	325 µs	102 µs	142 µs
5000 bytes	324 µs	364 µs	162 µs	202 µs

The analysis revealed that GSI's requirements are not fulfilled: the upper-bound delivery latency for the required size of CM and max distance of 2km is greater then 100 µs.

The solution to decrease delivery latency is targeted into the CD only and takes advantage of its characteristics (broadcast within a VLAN, sent by privileged node). We propose to break the highest priority of the CoS into two (unicast and broadcast) and use the highest priority broadcast Ethernet traffic only for the CD. Moreover, this particular traffic shall be forwarded using the cut-through method (unlike the store-and-forward method used normally in the switch) which can be effectively fast for the broadcast traffic with a single source (DM). The results, presented in Table 2 (*Cut-through* column), show a significant improvement. The solution requires hardware-supported cut-through forwarding in the switch as described in [6].

DATA RESILIENCE

Forward Error Correction

The objective of the FEC scheme is to decrease the loss rate of the CMs, preferably, to less then one per year. WR uses as a physical medium Fiber Optic and CAT-5. The number of received corrupted bits compared to the total number of received bits is called Bit Error Rate (BER). The value of BER characterizes a physical medium and can be used to characterize the entire switched network. A WRN can be seen as a Packet Erasure Channel (PEC) or as a Binary Erasure Channel (BEC) depending on the effect of a bit error on the frame. If the frame is lost (e.g. dropped by the switch due to a corrupted header or lost during switch-over between redundant components), the WRN is a PEC. If the bit error happens in the link between a switch and node, a corrupted frame can be used (optional) to attempt frame recovery. In such case, the channel is called BEC. Each type of channel requires a different FEC solution. Therefore two concatenated FECs are used in WR. Reed-Solomon (R-S) coding is used for the PEC and allows to encode k original-frames into n encoded-frames ($n > k$). Reception of any k encoded-frames can be used to decode the original frames. Hamming coding with additional parity (SEC-DED) is used for the BEC and allows to detect up to two simultaneous bit errors and correct a single error. These two schemes (R-S and Hamming) are combined to encode each CM – it is split into two and encoded using R-S into four messages (two original and two with redundant data). Each of the four messages is then encoded using Hamming. Such encoded messages are sent in a burst of 4 Ethernet frames. Reception of any two of these frames enables to decode the original Control Messages. A systematic analysis, using the BER characteristic of the WRN, proves that the presented FEC scheme guarantees less than single CM lost per year due to physical medium imperfection, as can be seen from Table 3.

Table 3: GSI and CERN FEC Characteristics

Parameter	GSI	CERN
Control Message length	500 bytes	1500 bytes
Control Message per year	$3.145 * 10^{11}$	$3.145 * 10^{8}$
Max Bit Correct.	1	1
Payload Length	294 bytes	854 bytes
Num Encoded Frames	4	4
Needed Frames to Receiver	2	2
Probability of Loosing a CM	10^{-14}	10^{-13}

Rapid Spanning Tree Protocol (RSTP)

In an Ethernet network with redundant topology, the problem of loops (causing "broadcast storms") is handled by the Rapid Spanning Tree Protocol (RSTP) . It creates a loop-free logical topology by blocking appropriate ports in switches, and unblocks them in case of topology break (due to element failure).

The functionality provided by the RSTP is essential for the WRN. However, the convergence speed provided by the standard implementation of the RSTP (milliseconds at best) would cause many CMs to be lost during the process. This is not acceptable, we need a solution which is fast enough to prevent loosing the CMs at all. Since we know the size-range of the CMs (Table 1) and how they are FEC-encoded into Ethernet frames, we can calculate the maximum value of the convergence time: 3 µs. This time is smaller than the duration of transmitting a single frame with FEC-encoded CM – this ensures that no more than two frames with FEC-encoded CM are lost, thus the CM can be recovered.

In order to achieve a convergence time of 3 µs, the switch-over between active and backup connections needs to be performed in the hardware as soon as the link-down is detected. It can only be done if the alternative topology is known in advance. The knowledge of alternative topology is translated into an RSTP-assignment of alternative and backup roles of switch ports, i.e at least one port with alternative role must be identified in every switch (except the topology-root switch). If we ensure, by restricting the topology, that RSTP identifies the alternative links, we can use its data to feed the hardware, consequently achieving the required convergence time and staying standard-compatible: the hardware switch-over is just a faster RSTP-driven convergence. The required topology restrictions, described in [6], greatly overlap with these imposed by the Time Distribution.

CLOCK DISTRIBUTION

A seamless switch-over between redundant sources of timing (uplink ports) is heavily supported by the Clock Recovery System (CRS) [2] of the switch and the WR extension to PTP (WRPTP)[3].

Figure 1 presents an example where a switch (timing slave) is connected (by its uplinks 1 & 2) to two other switches (primary and secondary masters) – the sources of timing. On both uplinks the frequency is recovered from the signal and provided to the CRS. Similarly, WRPTP measures delay and offset on each of the links and provides this data to the CRS. The modified Best Master Clock (mBMC) algorithm [3] decides which of the timing masters is "better" and elects it the primary, the other is considered secondary (backup). The information from *uplink 1* (primary) is used to control the CRS and adjust the local time. However, at any time all the necessary information from the *uplink 2* is available and a seamless switch-over can be performed in case of primary master failure [2].

In addition to the switch-over-related synchronization instability, the variation of external temperature can cause an accuracy degradation. This problem, however, is solved by the PTP standard itself. By frequent link delay measurements, the fluctuation is compensated.

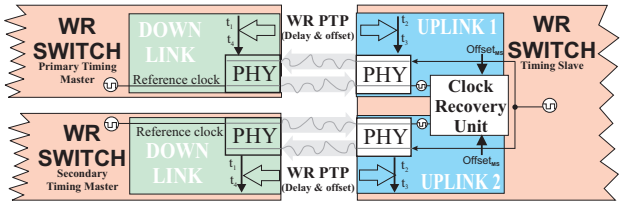


Figure 1: Seamless switch-over.

OVERALL RELIABILITY

The final equation of the WRN reliability is a sum of the data and clock distribution reliabilities. The clock distribution is assumed to be sufficiently accurate as long as there is a connection between the TM and all the nodes. The same applies to the CD distribution: as long as there is a valid connection, the FEC makes sure that the data is delivered with a sufficient reliability and the latency calculations prove it to be deterministic while the congestion is prevented by CoS and limited number of data sources (DM). Consequently, the overall reliability is strongly dependent on the WRN topology, which needs to be appropriate for the proposed solutions (SyncE, H/W-supported RSTP, upper-bound latency).

For the comparison of different network topologies, we consider the reliability of a network of switches. Each node is connected to such a network with M links (each to a separate switch). The value of M reflects the level of redundancy (M=1 for no redundancy, M=2 for double redundancy, etc).

In the calculations of the network reliability we used the idea of Mean Time Between Failure (MTBF) and its relation with the failure probability presented in [4] (a very simplified mathematical model). In order to calculate the MTBF of the entire network, we need the MTBFs of each network component: switches and links. Since the WR switches are still under development (no MTBF measured), we used representative values for CISCO switches (2, 10 and $100{*}10^4$[h]). Two estimation methods were used: "Fault Tree analysis" [7] and analytic. Both provide just rough estimations of the reliability. The former allowed to estimate two-terminal reliability (DM to single node) of simple non/double/triple-redundancy topologies (P_f). The most desired value is the all-terminal network reliability ($P_{f\_Network}$), where : $P_f < P_{f\_Network} < N_{nodes} * P_f$. Table 4 presents rough estimations of $P_{f\_Network}$ using analytic calculations for the three considered topologies ($MTBF_{Switch}$=200 000[h]). However, to meet the requirement of ≈2000 nodes and only three network layers (hops), the Data Master node is connected to more separate switches than the level of redundancy (M). The estimations show that a triple redundancy topology can barely satisfy the requirements by CERN (Table 1).

Table 4: WRN Topologies's Reliabilities

Redundancy	Switches	P_f	MTBF[h]
No	127	$2.08 * 10^{-3}$	$5.77 * 10^3$
Double	292	$4.71 * 10^{-7}$	$2.55 * 10^7$
Triple	495	$3.06 * 10^{-11}$	$4.08 * 10^{11}$

CONCLUSIONS

A WRN must be considered as an ordinary Ethernet network with extra optional built-in features which, when properly used, can make it robust and more reliable. This, however, comes at a price of topology restrictions and redundant elements (money). The reliability study described in this article and detailed in [6] presents areas which need to be addressed to increase the reliability of a WRN. The development of WR is an on-going effort and some of the suggested solutions have been already properly investigated or developed (FEC, clock distribution) while the others need further verification (RSTP, cut-through forwarding). Suggested solutions enable to fulfill the requirements set by CERN and GSI. However the costs might trigger double-checking and re-justifying of at least two of them: upper-bound latency by GSI and the number of CMs lost per year. The former requires additional development efforts to achieve the required 100 µs. The latter requires a high level of network redundancy (triple or more) which is very costly. Since the network topology and its reliability calculations turned out to be the greater factor in the overall system reliability, it is necessary to perform more precise calculations and simulations to verify the rough estimations. This might include different techniques (e.g. Monte Carlo simulations) but also more real-life use cases (i.e. of the network layout suggested in [8], which was not available at the time of described study). Especially, we need to take into account and include into calculations the fact that not all the nodes connected to the WRN are equally critical in real-life applications.

REFERENCES

[1] J. Serrano, P. Alvarez, M. Cattin, E. G. Cota *et al.*, "The White Rabbit Project," in *ICALEPCS*, Kobe, Japan, 2009.

[2] T. Wlostowski, "Precise time and frequency transfer in a White Rabbit network," Master's thesis, Warsaw University of Technology, may 2011.

[3] E. Cota, M. Lipinski, T. Wlostowski, E. Bij, and J. Serrano, "White Rabbit Specification: Draft for Comments," http://www.ohwr.org/documents/21, july 2011, v2.0.

[4] K. Dooley, *Designing Large-Scale LANs*. O'Reilly, 2002.

[5] White Rabbit. http://www.ohwr.org/projects/white-rabbit.

[6] C. Prados and M. Lipinski, "White Rabbit and Robustness," http://www.ohwr.org/documents/103, March 2011.

[7] "Reliability workbench, fault tree," www.isograph.com.

[8] J.-C. Bau and M. Lipinski, "White Rabbit CERN Control and Timing Network," http://www.ohwr.org/documents/85, July 2011.

STATUS OF THE RBAC INFRASTRUCTURE AND LESSONS LEARNT FROM ITS DEPLOYMENT IN LHC

I. Yastrebov, W. Sliwinski, P. Charrue, CERN, Geneva, Switzerland

Abstract

The distributed control system for the LHC accelerator poses many challenges due to its inherent heterogeneity and highly dynamic nature. One of the important aspects is to protect the machine against unauthorised access and unsafe operation of the control system, from the low-level front-end machines up to the high-level control applications running in the control room. In order to prevent an unauthorized access to the control system and accelerator equipment and to address the possible security issues, the Role Based Access Control (RBAC) project was designed and developed at CERN, with a major contribution from Fermilab laboratory. Furthermore, RBAC became an integral part of the CERN Controls Middleware (CMW) infrastructure and it was deployed and commissioned in the LHC operation in the summer 2008, well before the first beam in LHC. This paper presents the current status of the RBAC infrastructure, together with an outcome and gathered experience after a massive deployment in the LHC operation. Moreover, we outline how the project evolved over the last three years and give an overview of the major extensions introduced to improve integration, stability and its functionality. The paper also describes the plans of future project evolution and possible extensions, based on gathered users requirements and operational experience.

INTRODUCTION

In high energy and high intensity accelerators as the LHC, the energy stored in the beams is orders of magnitude above the damage level of accelerator components like magnets. Uncontrolled release of this energy can lead to serious damage of equipment and long machine downtimes. In order to cope with these potential risks CERN has developed a multi-pronged approach for machine safety that includes Role-Based Access Control (RBAC) [1] system. The main objectives of the project are: protection of accelerator equipment and control system against unauthorized access; definition of the operational access rules (who can do what and when) and providing facility to trace and audit the access requests.

The project was started in 2006 and was successfully commissioned and deployed for LHC in 2008, well before the first beam. Nowadays, RBAC protects all LHC equipment and selected experiments equipment against unauthorized access. Since 2008, the project has significantly evolved, following new operational requirements and collected users feedback.

RBAC ARCHITECTURE

Development of RBAC started in 2006, when the core part of the LHC control system was already in place. It was built on top of the existing software components and became an integral part of the Controls Middleware (CMW) [2] project. CMW provides communication infrastructure for all CERN accelerators, which is composed of centrally managed services (e.g. Directory Service) and middleware libraries (client/server) for all operational platforms. It implements two communication models: request-reply and publish-subscribe, organized in client-server relationships. By invoking the device access methods (i.e. *GET*, *SET* and *MONITOR*), clients can read, write and subscribe to device property values. Within this model, authentication (A1) is done on the client side and authorization (A2) is imposed on the server side by interception of the incoming client requests. The overall RBAC architecture including flow of an authentication token, accompanying every request, is illustrated in Figure 1.

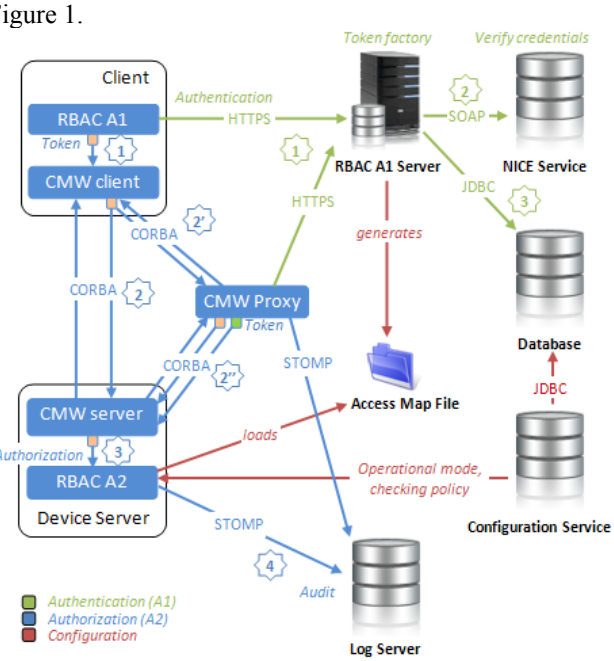


Figure 1: The overall RBAC architecture.

Configuration

Access rules are defined by equipment specialists for every device class. They are stored and managed centrally in the Controls Configuration Database [3]. Access map is a tab-separated text file, located on NFS, inside the CERN's technical network, which is in fact a snapshot of the database access rules for a given CMW device server. Every access map is digitally signed by the central RBAC A1 server to prevent against unauthorised modifications by third parties.

Authentication (A1)

The purpose of authentication is to verify the digital identity of a principal. If the authentication process succeeds, its result is a digitally signed authentication token that is returned to the application. The token is a short-term uniform substitute of the real credentials. It is issued by the central A1 authentication server that can reliably verify the users identity [3]. Authentication is a 3-step process: application sends request over encrypted HTTPS channel (1); authentication A1 server verifies credentials using remote, encrypted SOAP access to the NICE service (2) and then in case of success returns a token containing users roles and other details extracted from the Controls Configuration Database (3) [4].

Authorization (A2)

Authorization is the process of verifying that a known identity has an authority to perform a certain operation. Prior to authorization, a client application has to be authenticated (1). Following, it can use the obtained token to interact with various parts of the control system: access the equipment devices directly (2) or through a proxy (2') or another middle-tier server. Front-ends and the middleware library that are receiving such calls verify the token, thus confirming the identity of the remote party and can use it as a base for authorization (3) [5].

OPERATIONAL DEPLOYMENT IN 2008

During the deployment campaign in 2008, RBAC was successfully integrated in all layers of the LHC control system (see Figure 2).

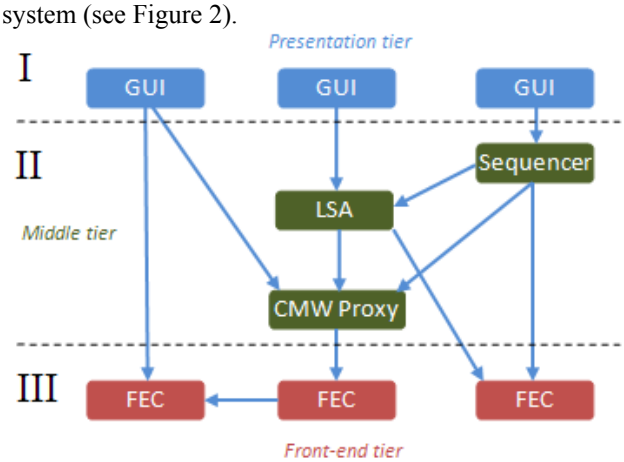


Figure 2: Layers of the Control System using RBAC.

In the presentation tier (I), RBAC authentication library was introduced for client applications running in the CERN Control Centre (CCC), where all CERN accelerators are controlled by the Operation team. For Java clients, RBAC provides login services, Spring (open-source, lightweight container for Java) beans and GUI components targeted for quick and easy integration of RBAC into existing applications.

In the middle tier (II), security components provided by RBAC were integrated in the high-level control systems (e.g. LSA, Sequencer) that help operators and physicists to commission, monitor and control the LHC machine. RBAC became an integral and core component of all major control subsystems including: LHC Software Architecture (LSA) [6], Software Interlock System (SIS) [7], Sequencer [8] and CMW Proxies.

In the front-end layer (III), RBAC integrates with CMW and Front-End Software Architecture (FESA) [9]. At this level the software is written in C++ and it uses RBAC A2 authorization library. Selected front-end machines might also use the A1 authentication service in order to communicate with protected equipment accessible through other front-ends.

To facilitate the introduction of RBAC into such a large and distributed system the algorithm of dynamic authorization was designed. It takes into account not only the access rules, but also an internal state of the subject under question. For RBAC, three different checking policies were introduced: NO-CHECK (no protection, no authentication required), LENIENT (access restriction for protected properties, authentication is optional) and STRICT (authentication and authorization is obligatory). They treat differently the access rules and have different requirements for the authentication token. For each front-end server the associated checking policy is stored in database, from where it is propagated to the running servers by the CMW Configuration Server and it can be changed at runtime from the administrative GUI application. The major advantage of this approach is the possibility to make a phased introduction of the access control in a large, distributed system as LHC.

During the deployment campaign in 2008 and 2009, the STRICT policy was enforced for all equipment systems in LHC and LENIENT policy for all other machines, i.e. injector chain (e.g. PS, SPS).

EVOLUTION OVER LAST 3 YEARS

Over the last 3 years the RBAC infrastructure has been significantly improved in order to comply better with the operational requirements and to provide seamless integration with all layers of the control system. In this section we list the most important changes in the project, which were introduced over the last 3 years.

Temporary Roles

In 2009, the concept of temporary roles was introduced. It brings the possibility to assign a certain role to some user (e.g. piquet role to an equipment expert) and grant him specific access rights but only during the limited intervention time. After that time the role is automatically revoked for that user.

Critical Roles

The Management of Critical Settings (MCS) [10] system is aimed to protect the most critical parameters in either potentially dangerous equipment or protection devices (e.g. Beam Loss Monitors, Collimators). It is complementary to the RBAC infrastructure. Both systems, RBAC and MCS, are fully integrated in the

control system for LHC and SPS and were successfully commissioned already before the first beam in LHC. In addition to RBAC's authentication and authorization MCS provides a mechanism to guarantee data integrity at all times using the digital signatures.

RBAC was extended to provide management of the public/private key pairs (stored in the private, local key store) as a part of the role management module. RBAC database keeps association of the critical MCS roles with the public/private key pairs. The private keys are always kept secret and never leave the RBAC A1 server. The digital signatures are generated by the RBAC A1 server using the private key from the key store. The public keys are made available to front-ends, namely to FESA servers and other applications, which verify the MCS signature.

For MCS roles several additional restrictions were implemented in RBAC: role lifetime constraints and limit of active critical MCS roles in a token.

Operational Mode

During accelerator shutdown or technical stop it is often necessary to allow access to machine for a wider range of users than during normal operation. In such cases an expansion of the access rules is not always desirable and appropriate. Firstly, it may weaken the system security and secondly it requires significant administrative costs. To address this problem notion of the *Operational Mode* was introduced to the access rules as an additional condition, which facilitated extensions of the access rules for non-operational interventions, while keeping operational access rules unchanged.

Support for CMW Proxy

CMW Proxy servers act as an intermediary layer between clients and actual front-end servers. A typical use-case for a proxy is to reduce the number of open connections to the front-end and to reduce the load that is associated with processing of subscriptions. When several clients subscribe on the same property, proxy establishes just one connection to the front-end and then broadcasts updates to all interested clients. The major security issue in this model is how to enforce the access control for subscriptions.

In order to address this problem authentication for CMW Proxies was introduced. At start-up, each proxy performs authentication by location, without using stored credentials and obtains a token that contains the 'CMW-PROXY' role. Access rules for devices working behind a proxy must allow the client subscriptions on desired properties for that role. This approach has several advantages. First, being very simple, efficient and non-intrusive it enforces access control in a single place. Second, it helps equipment specialists to impose usage of a proxy for certain devices thus preventing direct access and performance problems.

Virtual Devices

Virtual device is a special type of device that usually represents a unit of high-level business logic, which is not deployed on a physical front-end machine. A typical use-case for a virtual device is implementation of the high-level control parameters on top of existing physical devices. Since there is no front-end device server, which hosts a virtual device it is problematic to enable authorization. Therefore, RBAC framework was extended to provide a lightweight authorization scheme for any type of device, including both hardware and virtual. In this case authorization is performed by requesting RBAC A1 server that checks privileges for the supplied token.

QUALITY ASSURANCE

After introduction and commissioning of the new extensions, RBAC project team focused on test-driven development for C++ and Java components, in order to improve the overall quality and reliability. In the following paragraphs the detailed description of this process is presented.

Quality Assurance and Testing

A major effort was invested in preparing the tests for all core components. All tests can be split into 3 groups: *unit tests* that verify functionality of individual classes; *integration tests* for libraries that guarantee compatibility between different versions and *system tests* for the whole set of products. System testing verifies that the completely integrated system meets the requirements. For this, the setup called Testbed [11] was used, to test core control components before their operational deployment. For integration testing Bamboo server was used that provides the Continuous Integration environment. Each source code update in the core components triggers recompilation and execution of all unit tests. Next, code is analysed with the Clover tool that computes the code coverage and detects most risky and most complex classes and generates a detailed report. An example report is demonstrated in Figure 3.

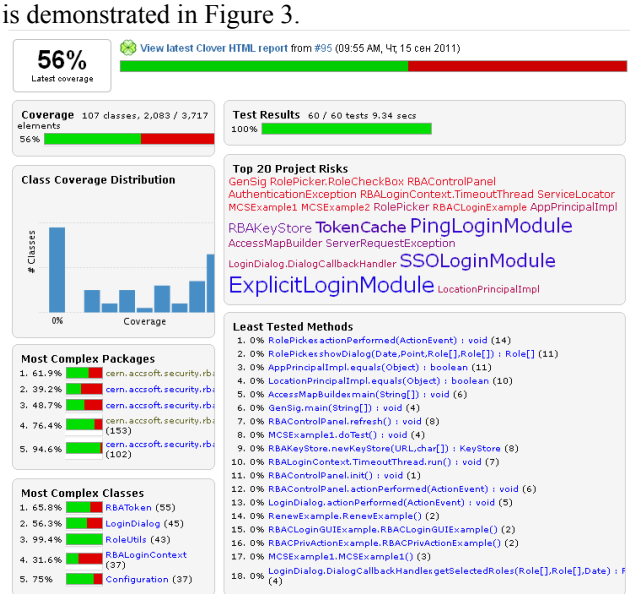


Figure 3: Example of the Clover report for RBAC.

Performance Improvements

Authorization check is performed for each access to front-end, therefore it should execute fast and should not hinder the performance of CMW. Having this in mind, the authorization algorithm was completely redesigned, which resulted in 10 times performance gain.

Codebase Refactoring

During last 3 years a major effort was invested into refactoring of the RBAC codebase. Most components were significantly redesigned and improved. The main aspects of refactoring: removal of code duplication, separation of interfaces from implementation, usage of dependency injection to reduce coupling and make the code more testable, split of heavy classes to make them more cohesive and reusable, code documentation.

For RBAC components written in C++ the new release system based on Maven [12] is used since 2011. It helped to standardise and to unify the development process by introducing versioning and dependency management.

STATUS OF THE PROJECT

The RBAC infrastructure was integrated with all layers of the CERN Control System and successfully deployed and commissioned in LHC.

Collected Experience

Lessons learnt from RBAC deployment in LHC:

1. Close collaboration with equipment experts and the Operation team in CCC was essential for successful deployment and operation of RBAC.
2. The crucial success factor was the staged deployment strategy: new functionality was tested during several 1-3 days LHC dry-runs, then verified during machine checkout and commissioning phases before the final deployment for beam operations.
3. Dynamic authorization based on several checking policies allowed for step-by-step introduction of RBAC without interrupting the running systems.
4. Several administrative tools had to be upgraded to support introduction of RBAC; to provide additional diagnostics and to allow changing checking policies for device servers in the runtime.
5. Better logging and diagnostics at all layers was required. This was a critical requirement for debugging and helped to solve many issues.
6. Database driven approach provided means for enforcing data consistency and simplified development of web-based administrative forms.
7. Configuration of access rules for sub-systems (e.g. BLM, BPM, Collimation) was delegated to equipment experts (reduced administration costs).

Future plans

This paragraph lists the major new features that are planned for implementation in the coming year:

1. Introduce Spring framework in RBAC server.
2. Make authentication server resistant to the database connection failures.
3. Revise the authentication by location mechanism and make it more secure.
4. Automate propagation of the access rules from database to the running device servers.

CONCLUSIONS

The RBAC project was developed at CERN, with a major contribution from Fermilab laboratory. It was successfully deployed and commissioned in LHC operations in the summer 2008.

The system successfully passed many centrally organized tests. The feasibility, performance and overhead of RBAC were experimentally evaluated. The results show that the overhead is acceptable and the chosen approach can be effectively used to enforce access control in the CERN control system.

Currently RBAC is used to protect all LHC equipment and selected equipment of other machines. Nevertheless, there are still few areas where current implementation can be improved and extended in order to expand the area of its applicability.

REFERENCES

[1] S.R. Gysin *et al.*, "Role-Based Access Control for the Accelerator Control System at CERN", ICALEPCS'07, Knoxville, Tennessee, USA.
[2] N. Trofimov *et al.*, "Remote Device Access in the New Accelerator Controls", ICALEPCS'01, San Jose, USA.
[3] A. Petrov *et al.*, "User Authentication for Role-Based Access Control", ICALEPCS'07, Knoxville, Tennessee, USA.
[4] Z. Zaharieva *et al.*, "Database Foundation for the Configuration Management of the CERN Accelerator Controls System", ICALEPCS'11, Grenoble, France.
[5] K. Kostro *et al.*, "Role-Based Authorization in Equipment Access at CERN", ICALEPCS'07, Knoxville, Tennessee, USA.
[6] G. Kruk *et al.*, "LHC Software Architecture (LSA) – Evolution toward LHC beam commissioning", ICALEPCS'07, Knoxville, Tennessee, USA.
[7] J. Wozniak *et al.*, "Software Interlocks System", ICALEPCS'07, Knoxville, Tennessee, USA.
[8] V. Baggiolini *et al.*, "A Sequencer For the LHC Era", ICALEPCS'09, Kobe, Japan.
[9] M. Arruat *et al.*, "Front-End Software Architecture", ICALEPCS'07, Knoxville, Tennessee, USA.
[10] W. Sliwinski *et al.*, "Management of Critical Machine Settings for Accelerators at CERN", ICALEPCS'09, Kobe, Japan.
[11] J. Nguyen Xuan *et al.*, "Testbed for Validating the LHC Controls System Core Before Deployment", ICALEPCS'11, Grenoble, France.
[12] J. Nguyen Xuan *et al.*, "A C/C++ Build System Based on Maven for the LHC Controls System", ICALEPCS'11, Grenoble, France.

DEPENDABLE DESIGN FLOW FOR PROTECTION SYSTEMS USING PROGRAMMABLE LOGIC DEVICES

M. Kwiatkowski*, B. Todd†, CERN, Geneva, Switzerland

Abstract

Programmable Logic Devices (PLD) such as Field Programmable Gate Arrays (FPGA) are becoming more prevalent in protection and safety-related electronic systems. When employing such programmable logic devices, extra care and attention needs to be taken. The final synthesis result, used to generate the bit-stream to program the device, must be shown to meet the design's requirements. This paper describes how to maximize confidence using techniques such as Formal Methods, exhaustive Hardware Description Language (HDL) code simulation and hardware testing. An example is given for one of the critical functions of the Safe Machine Parameters (SMP) system, used in the protection of the Large Hadron Collider (LHC) at CERN.

CERN is also working towards an adaptation of the IEC-61508 lifecycle designed for Machine Protection Systems (MPS), and the High Energy Physics environment, implementation of a protection function in FPGA code is only one small step of this lifecycle.

The ultimate aim of this project is to create generic techniques and methods applicable to any PLD based system requiring a rigorous implementation and verification.

CERN AND THE LHC

The Large Hadron Collider is the world's most powerful particle accelerator. To reach the new frontiers of physics a centre of mass collision energy of 14 TeV is needed, giving a stored beam energy over 100 times higher than in any other machine (Figure 1).

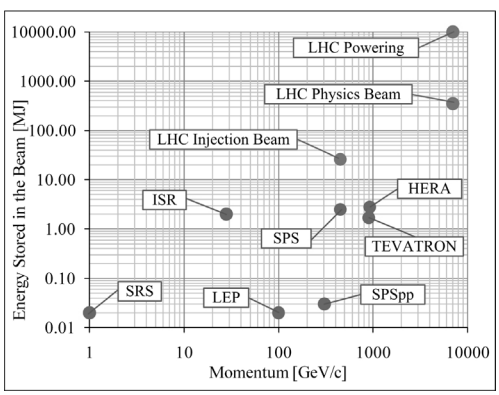


Figure 1: LHC Stored Energy versus other HEP machines [1].

* maciej.kwiatkowski@cern.ch
† benjamin.todd@cern.ch

The LHC is designed to accelerate two counter-rotating beams of $3.2 \cdot 10^{14}$ protons from an injection energy of 450 GeV to a collision energy of 7 TeV. At design values, the peak energy stored in each beam is equivalent to 362MJ, enough to heat and melt around 500kg of copper. A field of 8.3 Tesla is needed to hold the LHC beam in 27km circumference of the machine, this magnetic field is generated by 1232 super-conducting dipole magnets, each having a forward current of almost 13kA, being maintained less than two degrees above absolute zero (-273 degrees Celsius) in a bath of superfluid helium. At design energy, the total stored energy in the LHC magnet powering system is around 10GJ, and a loss of only $10^{-8} - 10^{-7}$ of the nominal beam [2] into one of the superconducting magnets will lead to a quench, where the magnet heats up, becomes resistive and must be switched off to prevent damage.

A complex MPS has been designed to mitigate the risks due to stored beam and magnet energy, a fundamental part of the MPS is the Safe Machine Parameters Controller (SMPC).

SAFE MACHINE PARAMETERS

For the correct protection of the LHC and its accelerator complex, several parts of the MPS require information about the machine's operational parameters. Values such as beam intensities, machine energies, squeezing factors, amongst several others, must be broadcast around the accelerator complex to correctly configure the MPS, and sent to the extraction interlock systems to ensure the correct interlocking of beam transfer between the Super-Proton Synchrotron (SPS) and the LHC.

These parameters are referred to as Safe Machine Parameters (SMP), as they must be generated and distributed

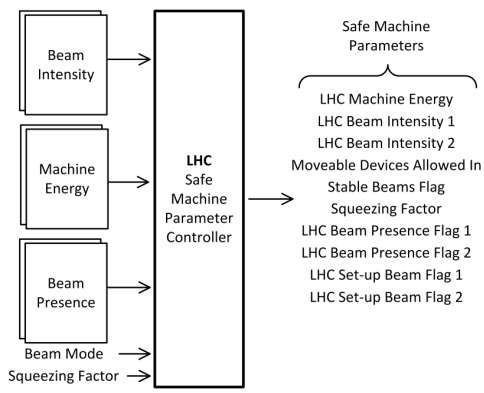


Figure 2: Safe Machine Parameters System.

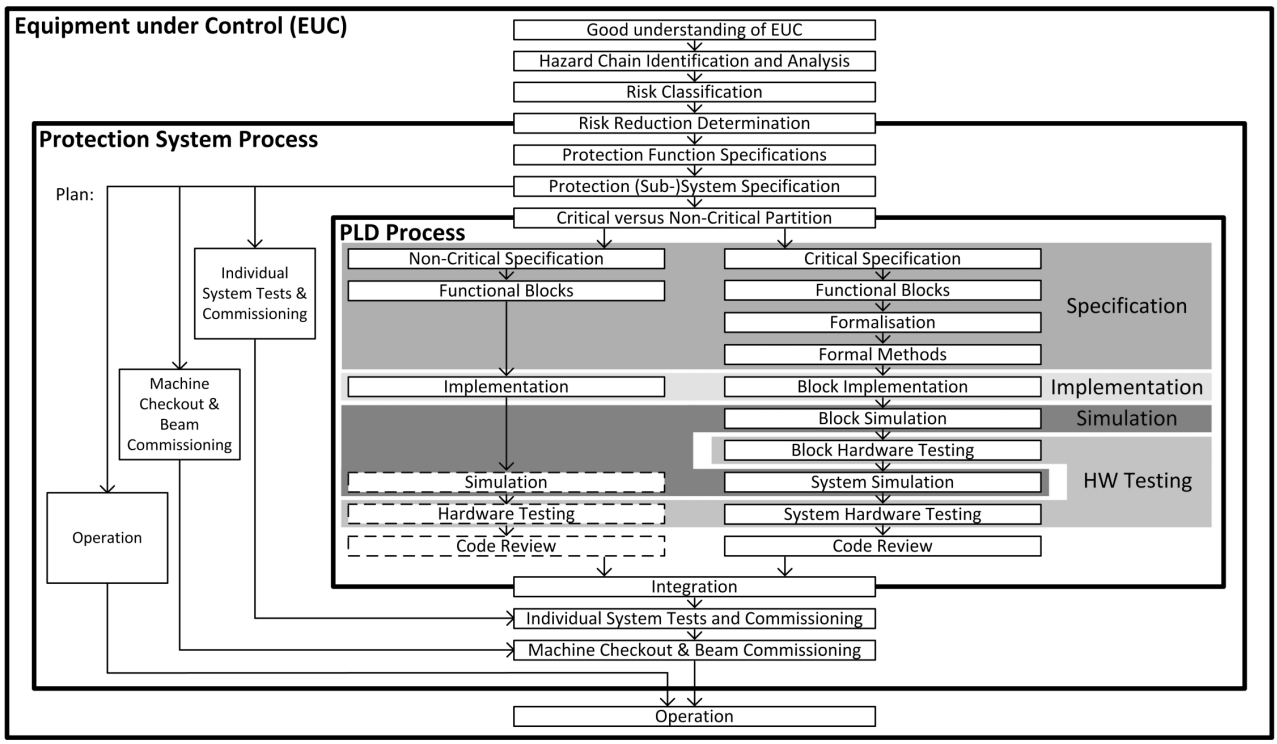


Figure 3: Machine Protection System Lifecycle Based on IEC-61508 E/E/PE Lifecyle.

around the accelerator complex with high dependability (safety, availability and reliability). The SMPC was developed to derive these SMP, taking information from several source systems and providing it to client system, as shown in the Figure 2.

The outputs of the SMPC either go directly to the extraction interlock controllers, or are broadcast around the machines using the General Machine Timing (GMT) network. These different parameters address two principle types of risk: risks during the extraction and transfer of beam from the the SPS into the LHC, and risks during LHC phases following the injection process.

The first group presents particularly tight requirements in terms of time and accuracy, as the transfer of beam from the SPS to the LHC is a very fast process, a single extraction of beam from the SPS is already capable of quenching and damaging LHC magnets.

The SMPC plays a key role in the protection of the LHC and its injector complex.

MACHINE PROTECTION SYSTEM LIFECYCLE

Requirements for the MPS, including the SMPC, came from decades of work and substantial investigations into the performance of the LHC and its MPS, following a deep-thinking argumentative approach. The risks due to LHC stored beam and magnet energy do not pose a threat to personnel or the environment, as as such are not a hard, legal requirement of the LHC project. Nevertheless, there are significant advantages to be gained by considering the development of the MPS as if it were a safety system, being legally required.

During the development of the SMPC, and some of the other MPS sub-systems, such as the Beam Interlock System and Powering Interlock Controllers, common themes and ideas have began to emerge. The combination of these ideas has led to the concept of a MPS Lifecycle (MPSL), based on the IEC 61508 overall system lifecycle. The MPSL concept allows a consistent approach to the evaluation of existing parts of the MPS, whilst at the same time providing a framework for the development of new protection systems. Figure 3 shows this proposed MPSL.

The part of the MPSL which is relevant to the programmable logic device flow is that related to the implementation of so-called Protection Functions (PFs) which are being established using devices such as FPGAs. The following sub-sections detail some of the key concepts of the MPSL, before concentrating on those parts relevant for PLDs.

Equipment Under Control

The Equipment Under Control (EUC) must be well understood by protection system designers, in CERN's case, the EUC can be considered as the particle accelerator and its associated mechanical, electric and electronic equipment. To realise such complex systems, experts from many domains must work together to establish potential hazards and the effects these have on the EUC. Based on this risk analysis approach, each risk should be assessed and classi-

fied. Risk being a combination of the probability of occurrence and the severity of a hazardous event and it is usually expressed qualitatively (qualitative risk analysis).

Protection System Process

The designers are then to conceive mitigations for these risks by working on two fronts: working to reduce the likelihood of the hazardous event, and by working to reduce the consequences of such an event were it to occur. The ultimate goal of this is to reduce risks to acceptable levels, another way is to use the phrase "As Low as Reasonably Possible" (the ALARP principle). This means that for each risk that is identified, a reduction is needed to achieve the ALARP level. The higher the risk the larger the required Risk Reduction Level (RRL).

Several approaches can be used to reduce risks, some of which result in the creation of PFs. It is possible that several risks can be mitigated by many PFs, and the PFs can be spread across many sub-systems.

In the case of the LHC MPS, one of these sub-systems is the SMP system, which is required to implement several PFs, by exploiting PLDs.

PROGRAMMABLE LOGIC DEVICE PROCESS

The PLD design process starts with clear separation of what is part of the PFs from other non-critical functions. Functions that are part of the PF are to follow this PSL process, whereas those functions that are non-critical are not needed to have the same levels of rigour. The conceptual separation of the critical from non-critical functions should be followed through to the hardware realisation. Hardware must not mix critical and non-critical functions! If no other choice exists, then non-critical functions which share common devices with critical ones must also be verified using the full protection system process.

In case of the SMPC each printed circuit boards has two separate FPGA devices for functions. The first, a control FPGA device, is tasked with critical functions, the second, a monitor device, records and supervises the operation of the control device and implements the non-critical functions. For example, the online monitoring of the SMPC, which is important but not critical for the protection of the EUC.

Specification Phase

Each critical function must be decomposed into functional blocks, ones which lend themselves easily to analysis and understanding. In addition, each must be capable of being readily specified using a formal language.

For example, the block diagram of the SPS Setup Beam Flag (SBF) is shown on Figure 4. This flag is used in the process of beam injection from the SPS to the LHC accelerator. The SPS SBF is a function of SPS beam intensity, to increase availability the intensity information is redundant,

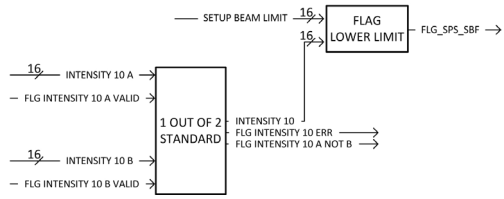


Figure 4: Setup Beam Flag blocks example.

being merged into one value. The one out of two block selects valid intensity value, when both of the sources are not valid the fail safe value is applied. The limit block compares calculated intensity value to a predefined threshold (SETUP_BEAM_LIMIT).

The formal specification of the SPS SBF calculation is presented on Figure 5. Error and selection flags (FLG_INTENSITY_10_ERR, FLG_INTENSITY_10_A_NOT_B) are not formalized because they are used for the monitoring purpose and they do not belong to the critical functionality. The key strength of this Predicate Logic formal language is in interpretation: if correctly written, it can only be understood by designers in a single way, which is not always the case for a traditional specification. The formal language also allows a formal verification of some design parameters, allowing a mathematical verification the completeness and the consistency of the specification.

```
FAIL_SAFE_HIGH_INTENSITY : int := 6.5535 * 10^14;

SPS_SMPC_SBF (
   SETUP_BEAM_LIMIT : int,
   INTENSITY_10_A : int,
   FLG_INTENSITY_10_A_valid : bool,
   INTENSITY_10_B : int,
   FLG_INTENSITY_10_B_valid : bool,
   FLG_SPS_SBF : bool
   ) :=
      exists INTENSITY_10 : int. (
      (INTENSITY_10 =
         if FLG_INTENSITY_10_A_valid then INTENSITY_10_A;
         if else FLG_INTENSITY_10_B_valid then INTENSITY_10_B;
         else FAIL_SAFE_HIGH_INTENSITY;
      )
      AND
      FLG_SPS_SBF =
         if INTENSITY_10 <= SETUP_BEAM_LIMIT then TRUE;
         else FALSE
   );
```

Figure 5: Setup Beam Flag in Predicate Logic.

Implementation Phase

Each of the smaller functional blocks which were conceived in the previous step are then implemented. Each block should be the correct size as to be simulated completely before being integrated into the sub-system. When all the blocks are implemented and have passed simulation, the complete system can be composed by connecting the blocks together. The actual implementation phase is a very small part of the overall time spent on PLD firmware development. The vast majority of the time is spent on simulation and testing. Hardware Description Language

(HDL) code, such as VHDL or Verilog, is written to describe PLD function, HDL languages should not be associated with programming languages but rather with modeling tools. The designer should always know what the expected result of the synthesis process is to be, this should be also verified using the output of synthesis tools. This kind of verification is much easier when the blocks are small and readily understood.

Software Simulation Phase

Software simulation is carried out on the block level as well as on the system level. Simulation with code coverage is a fundamental requirement for critical functions. A software test-bench is required, which should wrap the Unit Under Test (UUT) inside Bus Functional Models (BFMs), passing stimuli to the UUT and recording its responses. Behaviour which is not specified for the block or the subsystem can be detected and fixed, at the same time the test-bench should evolve to include new conditions as the weaknesses in code coverage are identified. It is preferred that a critical function achieves full code coverage.

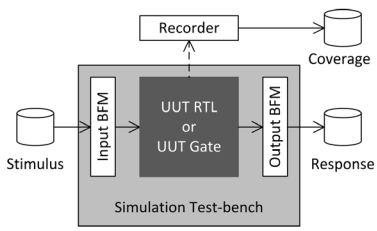


Figure 6: Software simulation with code coverage.

Hardware Testing Phase

Once the smaller blocks have been implemented, simulated and combined, the real hardware can be generated. This is then tested using a dedicated hardware tester. Hardware testing is obligatory on the system level but optional at the block level, this is due to technical difficulties in realising good test equipment and also the cost of the hardware tester construction must be considered. On the other hand, a complex block may warrant the investment of time and money in a dedicated test hardware.

The hardware tester generates input stimulus and checks the response of the Device Under Test (DUT), in much the same way as the simulation test-bench, but this time using real signals, logging real results. Its advantage over the software simulation is its speed and the possibility to introduce real distortions, such as noise on a signal. A very useful hardware testing tool provided by device manufacturers is an Embedded Logic Probe (such as Xilinxs ChipScope or Alteras SignalTap). The probe is integrated with the design and finally programmed into PLD DUT. It uses device memory resources for recording selected internal or external signals, thus the critical device must have spare cells and memory to accommodate this. With this analyzer in

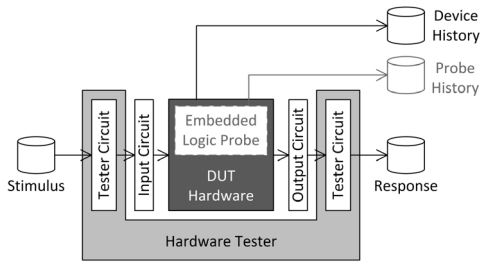


Figure 7: Hardware tester.

place it is possible to record internal signals using a variety of trigger conditions. A typical setup with an embedded logic probe is shown on Figure 7.

Code Review

Finally, once all elements have been proven to function as expected, the HDL code should be reviewed via at least an internal audit, and preferable including an external audit too. A key element during this phase is good documentation and especially a clear depiction of the formalised version.

CONCLUSIONS

All the phases described above should be documented and kept to form a basic safety case. When bugs or mistakes are found it will help to identify weak points of the test cases, testing processes, and even weaknesses in the design specification. All of this information can be used to correct future implementations. In addition, a clever selection of the design partitioning, and basic functional block requirements vastly increases HDL code reusage: thus the invested time in test benches and verification is also saved, and it is possible to reuse the same well verified blocks in the future designs.

The MPSL presented gathers many elements which have been often used in the development of systems at CERN, but are not necessary currently being done in a complementary and systematic way. These concepts and ideas are very much a work in progress, at the same time it is evident that such an approach encourages both designers, and those that specify designs to use good practices, and really understand what they require of a system. In doing this the confidence in the final system can be increased.

REFERENCES

[1] R. Assmann et al, "Requirements for the LHC collimation system", EPAC'02, La Vilette, Paris, 2002, http://jacow.org/e02/TALKS/TUAGB001.pdf.

[2] R. Schmidt and J. Wenninger, "Protection against Accidental Beam Losses at the LHC", PAC'05, Knoxville, 2005, http://jacow.org/p05/PAPERS/MOPA005.PDF.

RADIATION SAFETY INTERLOCK SYSTEM FOR SACLA (XFEL/SPRING-8)

M. Kago#, T. Matsushita, N. Nariyama, C. Saji, R. Tanaka, A. Yamashita,
JASRI/SPring-8, Hyogo, Japan
Y. Asano, T. Hara, T. Itoga, Y. Otake, H. Takebe, H. Tanaka,
RIKEN/SPring-8, Hyogo, Japan

Abstract

SACLA (SPring-8 Angstrom Compact free electron Laser), which comprises an 8 GeV linear accelerator and undulators, was constructed. Its radiation safety interlock system was designed to protect personnel from radiation hazards. This system controls access to an accelerator tunnel and monitors safety devices. It controls permission signals for accelerator systems in accordance with safety conditions. When a safety condition is not satisfied, the system turns off the permission signal, thereby terminating the electron beam. In particular, when the bending magnets controlling the beam transport route are not properly excited, the system must terminate the beam within 16.6 ms. Therefore, as part of this system, a beam route interlock was constructed to rapidly evaluate the correctness of the beam route. We developed a new optical module to transmit the permission signal at high speeds, over a long distance. The system achieved a response time of less than 6.7 ms.

INTRODUCTION

SACLA was constructed at the SPring-8 site. It generates high-intensity X-ray lasers using an 8 GeV linear accelerator and undulators. Beam commissioning began at the end of February 2011 [1]. The layout of SACLA is shown in Figure 1. The electron beam is generated by an electron gun (GUN) with a repetition rate of up to 60 Hz. The beam is accelerated to 8 GeV by the following series of RF accelerator cavities: pre-buncher (238 MHz), booster (476 MHz), L-band (1428 MHz), S-band (2856 MHz), and C-band (5712 MHz). The accelerated beam is switched by a switching magnet into two beamlines: BL1 and BL3. The beam is injected into the undulator to generate X-ray lasers that are transported to the experimental facility. Finally, the electron beam is bent toward a beam dump by a dump magnet and then safely disposed.

To protect personnel from radiation hazards, the radiation safety interlock system (safety interlock system) was designed and constructed [2] [3]. The safety interlock system manages access control and monitors safety devices. It controls permission signals for accelerator systems in accordance with safety conditions. When permission signals are turned off, the operations of the GUN, beam chopper, and RF are inhibited.

The safety interlock system needs to be reliable and stable. In addition, the system requires specific functionality, as described below.

First, fast beam termination is required. When the electron beam deviates from the beam dump, the electron beam generates unexpected radiation. Therefore, when this situation is detected, the next beam injection should be avoided, i.e., the safety interlock system needs to terminate the electron beam within 16.6 ms.

The other specific functionality concerns the transmission line of the permission signal. SACLA is a large facility whose total length is approximately 700 m. The accelerator section is approximately 400 m long. Within the klystron gallery of this section, 73 RF systems are distributed and placed. Thus, the permission signal must be transmitted over a long distance to a large number of accelerator systems. Moreover, to achieve fast beam termination, a high transmission speed is required.

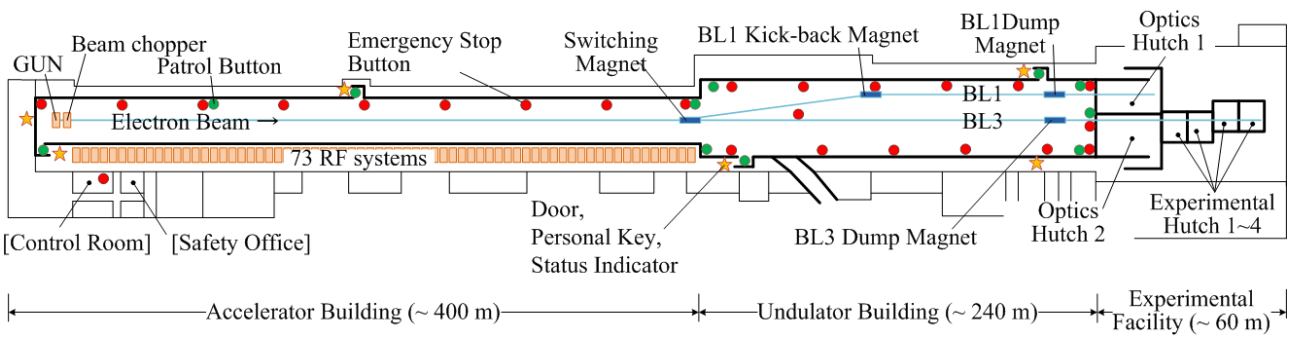


Figure 1: Layout of SACLA and the equipment related to the radiation safety interlock system.

#kago@spring8.or.jp

SAFETY INTERLOCK SYSTEM

Overview

We constructed the safety interlock system which consists of the following three interlock systems and a permission transmission system: a central interlock system (CIS), an emergency interlock system (EIS), and a beam route interlock system (BIS). Figure 2 shows a schematic view of the safety interlock system.

The CIS supervises the doors of the accelerator tunnel, personal keys, patrol buttons, status indicators, the safety signal from the beamline interlock system, and other safety systems, such as the radiation monitoring system. The CIS provides the access control to the accelerator tunnel, search confirmation, and indications about the accelerator operation.

The EIS monitors the emergency stop buttons. When an emergency stop button is pushed, all permission signals are turned off.

The BIS supervises the current of the electromagnets. To ensure that the electron beam is discarded into the beam dump, the BIS evaluates the beam route based on the excitation status of four electromagnets: the BL1 and BL3 dump magnets, the switching magnet, and the kickback magnet. If the dump magnet is not properly excited, the electron beam deviates from the beam dump core. In this case, the BIS turns off the permission signal for the GUN and the beam chopper within 16.6 ms.

The permission transmission system transmits the permission signal to the GUN, beam chopper, and 73 RF systems distributed over 400 m.

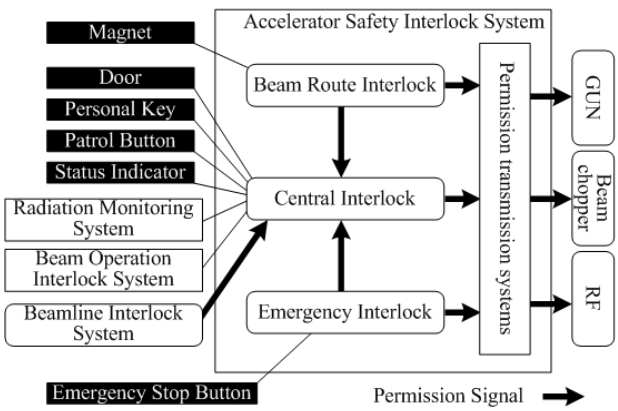


Figure 2: Schematic of the radiation safety interlock system.

The safety interlock system is required to be reliable and stable. At the SPring-8 site, programmable logic controllers (PLCs, Mitsubishi) have been used to construct interlock systems for 5 accelerators and over 50 beamlines [4]. PLC networks (MELSECNET/H) have also been utilized to input/output status signals from distant locations. We have operated them for over 14 years. PLCs present very few problems during their operation. Based on our experience, PLC-based systems are highly reliable and stable. Thus, to take advantage of these attractive characteristics, we based the CIS, EIS, and BIS development on PLCs.

Beam Route Interlock System

The BIS requires fast beam termination. However, this is difficult to achieve using PLC networks because the communication time is slower than the required response time. In addition, the PLC processing time depends on the size of the ladder program. Thus, we designed the BIS shown in Figure 3.

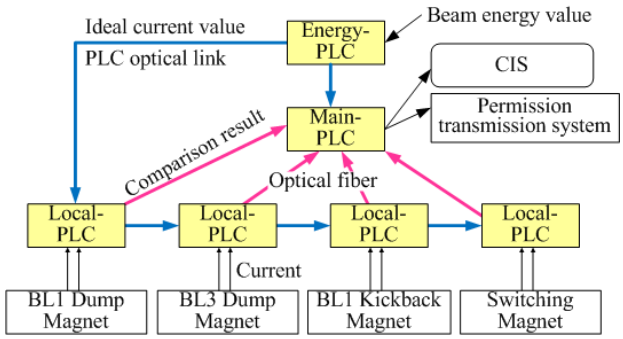


Figure 3: Schematic of the beam route interlock system.

We used several PLCs: Energy-PLC, Main-PLC, and Local-PLCs. The Energy-PLC and Main-PLC were installed in the safety office. The Local-PLCs were installed in the rack located near the power supply of the magnet approximately 600 m from the safety office. An optical PLC network used for data communication in slow control applications was used to connect the PLCs. The Energy-PLC executes high-load tasks, such as calculating the ideal currents for the magnets. The ideal current values are delivered to the Local-PLC via the PLC network. The Local-PLC inputs an analog signal proportional to the current of the magnet and compares the ideal value with the measured value. Then, the comparison result is sent to the Main-PLC. Because PLC networks cannot transmit signals within a few milliseconds, we used a direct fiber optic I/O connection between the Local-PLC and Main-PLC. The Main-PLC evaluates the beam route based on the comparison results and generates the permission signal.

We measured the response time of each device. The response times of the Main-PLC and the Local-PLC were both less than 3.0 ms.

Permission Transmission System

The permission signals for the GUN, beam chopper, and RF systems are generated by each interlock system. To achieve fast beam termination, the permission signals for the GUN and beam chopper must be stable and must have fast transmission speeds. Conversely, the permission signal for the RF systems is required to transmit over long distances and to a large number of systems. Therefore, PLC networks or metal cables cannot be used in the transmission line.

Thus, we developed a new optical module to transmit the permission signals and used it to construct the permission transmission, as shown in Figure 4. There are three permission transmission lines for the GUN, beam chopper, and RF systems. Permission signals are sent from the CIS, first, to module (A), and then, to module (C) via module (B). Module (C) outputs the signals to the target devices. Permission signals are transmitted from the EIS and BIS by inputting them to module (B) of the GUN and RF systems.

Two optical-fiber cables were used to connect the modules and ensure signal transfer. One of the fibers was used to send a static signal and the other was used to transmit the pulse signal (1 kHz) of the heartbeat. The transmission line for the RF systems used a total of 76 modules, and the total length of the optical fiber was more than 1.5 km.

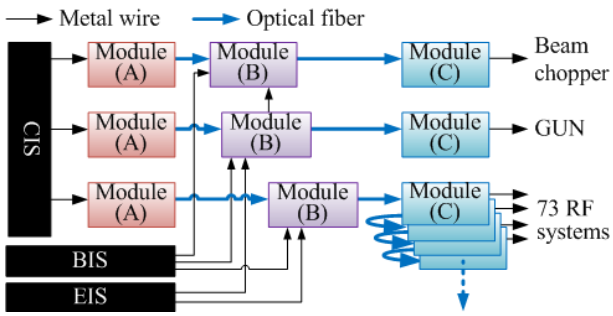


Figure 4: Schematic of the permission transmission system.

Figure 5: The newly developed module with a rack-mount unit of size 2U.

Figure 5 shows the developed module. A complex programmable logic device (CPLD) was utilized to reduce the processing time. To avoid noise, the circuit board was carefully designed. The modules are connected in a daisy-chain scheme and the distance between modules can be extended to up to 1 km. In the case of RF systems, the measured transmission delay time was less than 0.3 ms.

To verify reliability and stability, we conducted several tests, such as the electrical fast transient/burst test (international standard IEC 61000 4-4) and the long time operation test in the SCSS test accelerator [5]. The module passed all the tests.

MEASUREMENT OF RESPONSE TIME

We tested the total response time of the BIS and the permission transmission system. We chose the Local-PLC for the BL3 dump because it had the longest wiring from the Main-PLC. We measured the response time from the interlock action of the Local-PLC to outputting a permission signal from the module for the beam chopper. Figure 6 shows the 10,000 measurements obtained. The slowest response time measured was 6.7 ms, clearly satisfying the target value of 16.6 ms.

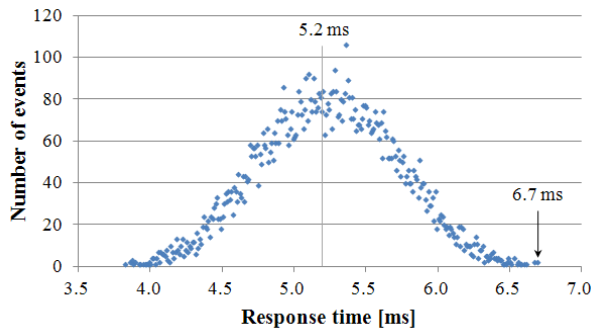


Figure 6: Total response time of the BIS and permission transmission system. The average response time was 5.2 ms with a standard deviation of 0.8 ms. The slowest value measured was 6.7 ms.

CONCLUSION

We constructed a radiation safety interlock system for SACLA. To stop the electron beam injection within 16.6 ms, an interlock system using multi-PLCs and an optical module were developed. The optical module was able to transmit a permission signal over a long distance to numerous accelerator systems. The radiation safety interlock system achieved a response time of less than 6.7 ms. The operation of the system commenced at the end of February 2011.

REFERENCES

[1] H. Tanaka, "Status Report on the Commissioning of the Japanese XFEL at SPring-8," IPAC'11, Spain, September 2011.

[2] N. Nariyama, et al., "Concept Of Radiation Monitoring And Safety Interlock System for XFEL/SPring-8," Proceedings of IPAC'10.

[3] M. Kago, et al. "Design of the Accelerator Safety Interlock System for the XFEL in SPring-8," ICALEPCS '09, Japan, October 2009.

[4] C. Saji, et al. "Upgrade of the Accelerator Radiation Safety System for SPring-8," ICALEPCS '09, Japan, October 2009.

[5] T. Shintake, "Status of Spring-8 compact SASE source FEL project," NIMA21188, Nuclear Inst. and Methods in Physics Research.

TANGO CONTROL SYSTEM MANAGEMENT TOOL

P. Verdier, J.L. Pons, F. Poncet, ESRF, Grenoble, France
N. Leclercq, Soleil, Gif-sur-Yvette, France

Abstract

TANGO [1] is an object oriented control system tool kit based on CORBA initially developed at ESRF[2]. It is now also developed and used by Soleil, Elettra, Alba, Desy, MAX-Lab, FRM II and some other labs. The TANGO concept is a fully distributed object oriented control system.

That means that several processes (called servers) are running on many different hosts. Each server manages one or several TANGO classes. Each class could have one or several instances (call devices).

This poster will show existing tools to configure, survey and manage a large number of TANGO components.

GOAL

The first goal is to know at a quick glance, if everything is OK in a control system, and otherwise to be able to diagnose a problem and solve it.

The second goal is to configure the control system and its components.

The third goal is to have long term analysis on components (logs, statistics, usage,....)

PRINCIPLE

On each host to be controlled, a device server (called Starter[3]) takes care of all device servers running (or supposed to be running) on this machine. The controlled server list is read from the TANGO database. A graphical client (called Astor[4]) is connected to all Starter servers and is able to:

- Display the control system status and component status using coloured icons,.
- Execute actions on components (start, stop, test, configure, display information, …)
- Execute diagnostics on components.
- Execute global analysis on large number of crates or database.

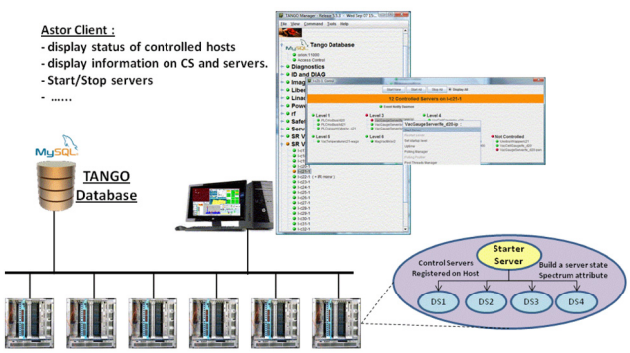


Figure 1: Astor/Starter principle.

STARTER DEVICE SERVER

This Tango device server, written in C++, is dedicated to control system management. An instance of the Starter must be running on each host in the control system.

Controlled Servers Status

When a Tango device server starts, it registers itself automatically in the TANGO database as running on specified host. At start-up, the Starter takes this list of servers registered on the host .

Periodically, a system call gets the list of running processes. For each controlled server, the Starter checks if the server is the running processes list. If it is running a second check is done by a ping on the administrative device to know if the server is alive. And finally a list containing the server names and their status is built.

Starting Servers

For each server, a control level can be defined. Level 0 means not controlled. At start-up, the Starter takes a list of servers registered on the host and its control level. Starting at level 1 to level n, if the specified server is not already running, the Starter tries to start it.

In order to not overload the database, the servers are started sequentially. To start the next server, one of the following conditions must be met:

- Server is alive.
- Server has failed.
- Time out.

A server can be started at any time on a command from a client. Using the same manner, a client command is able to start all servers having the same control level.

It is possible to configure the starter, using a property, to re-start a server automatically if it fails. This property defines a minimum time that has be running, to ensure that this server is able to run a bit.

Implementation

The Starter device server is supported for 3 operating systems:

- Windows.
- Linux
- Solaris

One of the main problems has been the different API's for system calls. Mainly to get running processes.

ASTOR – CLIENT TANGO MANAGER

A graphical client, using the java swing frame containing a tree to display the control system status.

On top of the tree we can find:

- The database status
- The access control status (if activated).

FEATURE OVERVIEW

Control System Status in a Quick Glance

The host status read from the Starter device server is displayed on a tree using coloured icons.

- Green if all controlled servers are running
- Blue if at least one server does not respond (start-up phase)
- Orange if at leas one controlled server is stopped
- Red if the starter itself does not respond.

The hosts can be grouped under branches by usage, operating system or whatever.... The status of the hosts under a branch is reported to the branch node.

Executing Actions on Tango Device Servers

A panel can be opened on a crate for more details. The status of each server is also displayed by a coloured icon.

- Green if the server is running
- Blue if the server is running but not responding.
- Red if the server is not running.

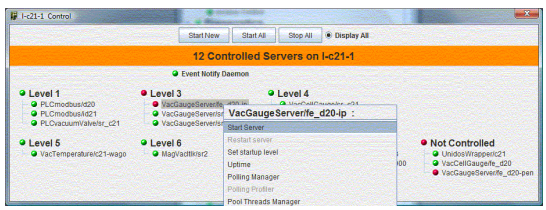


Figure 2: Host panel for server management.

By a right click on a server, a pop up menu is displayed to have an action or information on this server.

- Start/Stop servers.
- Test devices (single command or attribute).
- Check device states.

With one of the top buttons, you can start or stop all controlled servers registered on the specified host.

Using the same approach you can start a new device server. It will become a controlled server after you have chosen the control level.

On the host panel (Figure 2), the displayed status, is the server status, and not the device status, as most of the people belong. The specified server is able to serve many devices, and a simple icon cannot represents a combination of several status.

To know the status of each device select the item “Check States” and a panel showing devices status, sorted by classes.

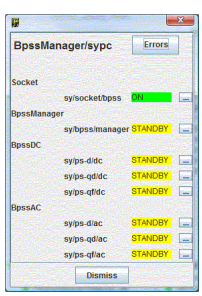


Figure 3: Device status panel.

Configure and Display Information

A database browser (called Jive) can be launched to edit device or class properties (see Figure 4).

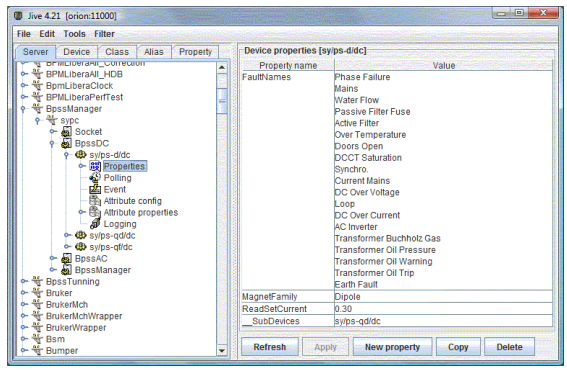


Figure 4: Jive (database browser).

It is also used to have information as to where a server is running, on which port, uptime,... Or to add new components, test devices (see Figure 5) and start a generic panel (called ATKpanel) to monitor devices (see Figure 6).

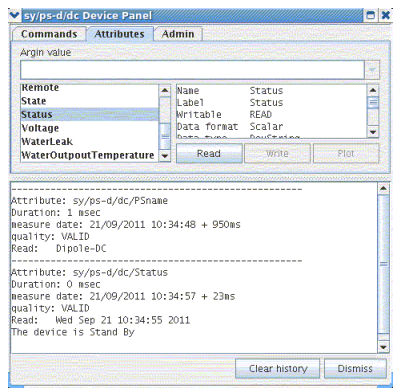


Figure 5: Device test panel.

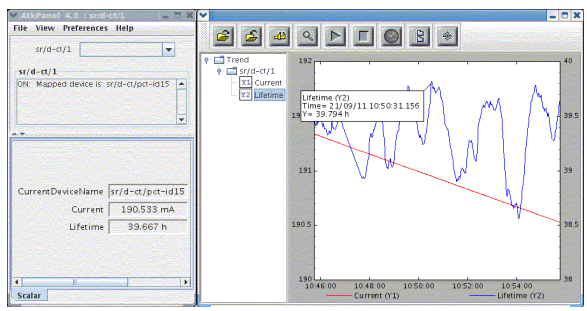


Figure 6: Device monitoring panel.

A wizard is available to guide the user to create a new server and to set class and device properties

Jive is able to set the polling period for device attributes and to set event parameters.

Help to Debug

To help debug a device, Jive and Astor propose a viewer (called LogViewer) to display on a specified device several outputs levels (see Figure 7). These outputs are time stamped and can be filtered to improve analysis.

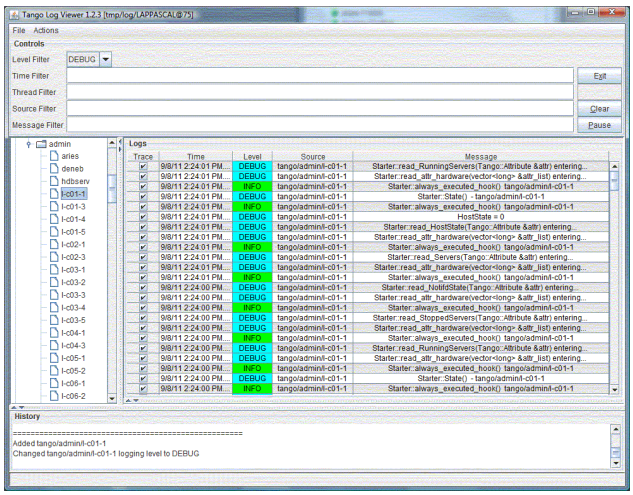


Figure 7: LogViewer (device outputs).

Attribute Polling Configuration and Tuning

A Tango device attribute can be polled periodically. To configure this polling, a diagnostic tool is available. It displays when each attribute has been polled the last four times (see Figure 8) and how much time was required to read and set it. It is very useful to tune the device attribute polling when a server has several devices with many attributes each. Or in case of a hardware problem with time out in reading.

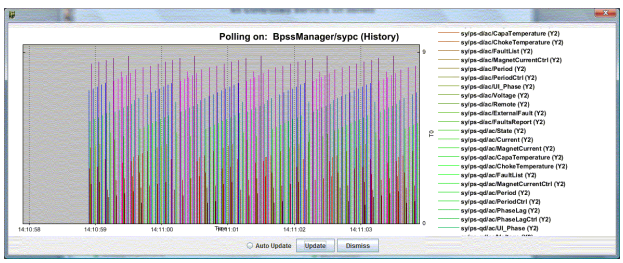


Figure 8: Polling profiler.

By default, a single thread is started to poll attributes. In case of several devices, a pool of threads is available. Astor proposes a graphic tool to distribute device(s) by thread as shown in Figure 9.

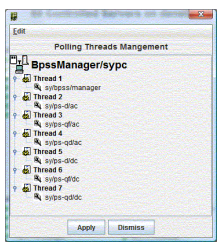


Figure 9: Pool of threads management.

Events Configuration and Test

Tango communication between client and server, could be done on events. A client subscribes to on event (e.g. an attribute change value) and will be waken up when this event will be pushen by the device. To configure and test events , Astor proposes a tool able to browse database to select device attribute, configure event parameters (period, change value, …) and start a test to check if events are received as expected (see Figure 10).

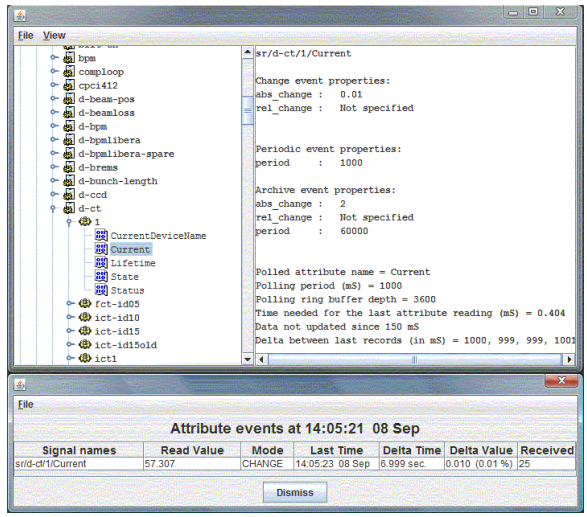


Figure 10: Event configuration and test panel.

Server Statistics

When a server, started by the Starter, is stopped by another way than a command on the Starter (kill signal, core dump, ...), this event is logged in a file. When the server is re started by the Starter, this information is logged too, Astor proposes to get information from this file for one host (see Figure 11) or for all controlled hosts, and compute statistics an availability for servers (see Figure 12).

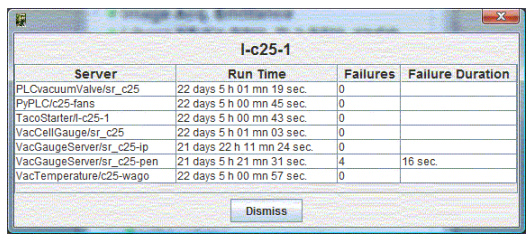


Figure 11: Statistics on a host.

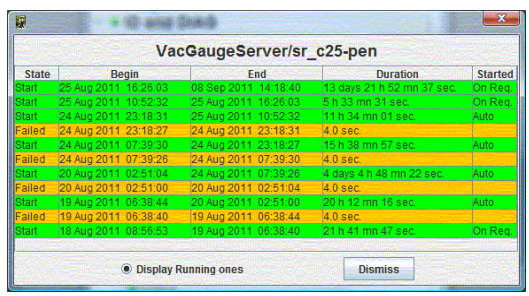


Figure 12: Statistics on a server.

Miscellaneous

Astor propose also:

- Simple logging about start/stop servers
- A panel to send command on several servers.
- A panel to show server usage (useful for large number of instance).
- A panel to configure control access if it is activated.

CONCLUSIONS

The couple Starter/Astor and related tools, are very useful to manage a large number of servers running on several hosts distributed around an accelerator or a large experimental physic control system. They help to configure diagnose problems, and for doing analysis and statistics.

REFERENCES

[1] http://www.tango-controls.org/
[2] http://www.esrf.eu/
[3] http://www.esrf.eu/computing/cs/tango/tango_doc/tools_doc/astor_doc/index.html
[4] http://www.esrf.fr/computing/cs/tango/tango_doc/ds_doc/tango-cs/System/starter/

FLYSCAN: A DISTRIBUTED FAST ACQUISITION SYSTEM FOR MULTI-DETECTOR EXPERIMENTS

F. Langlois, A. Buteau, X. Elattaoui, C. M. Kewish, S. Le, N. Leclercq, P. Martinez, K. Medjoubi, S. Poirier, A. Somogyi, SOLEIL Synchrotron, L'Orme des Merisiers, 91190 Gif-sur-Yvette, France
A. Noureddine, Médiane Système, 54 route de Sartrouville, 78232 Le Pecq, France
C. Rodriguez, GIST, 40 av André Morizet, 92513 Boulogne-Billancourt, France

INTRODUCTION

An increasing number of synchrotron beamlines need to use several detection techniques in parallel, and in fast acquisition modes. For example, a scanning microspectroscopy beamline could simultaneously collect data from two-dimensional (2D) position-sensitive X-ray detectors, one-dimensional (1D) energy-dispersive fluorescence analyzers, and several single-element beam intensity monitors.

Continuous scanning modes for fast data collection are required for scientific experiments conducted on such beamlines. That is, measuring signals from all detectors "on-the-fly" while actuators are moving, in order to avoid significant time overheads that are introduced by motor settling times in step-scan mode, and allow an experiment to be completed within a reasonable time frame. The implementation of this so-called "flyscan mode" is made possible by the progress in detector technology and the high photon fluxes available from 3rd generation synchrotron sources such as Soleil. The large amount of data produced in such an experiment has to be managed, either off-line and on-line, using software tools to analyze the data quality during collection and for categorizing and retrieving information for further scientific analysis.

Relying on our Tango [1] based software distributed architecture, a set of C++/java libraries, Tango devices and GUI applications has been developed to fulfill these requirements: (*i*) modularity, in terms of the number and type of detectors; (*ii*) performance, to handle these large data volumes; and (*iii*) data management, to store and retrieve information in a coherent way.

Taking as an example the challenging data acquisition demands of the future Soleil beamline Nanoscopium, this paper will present the status of development, deployment and operation of this "distributed fast acquisition system for multi-detector experiments".

THE SCIENTIFIC CASE OF THE NANOSCOPIUM BEAMLINE

The Nanoscopium beamline of Soleil will be dedicated to state-of-the-art scanning X-ray imaging techniques in the 5 – 20 keV range. It will provide unique research opportunities by combining the analysis of sample chemistry, *via* X-ray fluorescence and absorption spectroscopy, with structural analysis from coherent diffractive imaging at high spatial resolution (= 30 nm) in two and three dimensions. Elemental distributions and oxidation states can be quantified at the trace level in geological and biological samples for most of the elements starting from Ti. Fluorescence spectroscopy will target elements as light as phosphorous. The beamline is in the construction phase and will be operational in 2013.

These different detection schemes are to be used in flyscan mode in order to measure statistically relevant sample areas (up to several mm^2) and to perform tomographic scans without significant time overhead. Fast data collection will also result in reduced degradation of the cryo-cooled samples during the imaging experiments. Figure 1 shows a typical simultaneous detection mode.

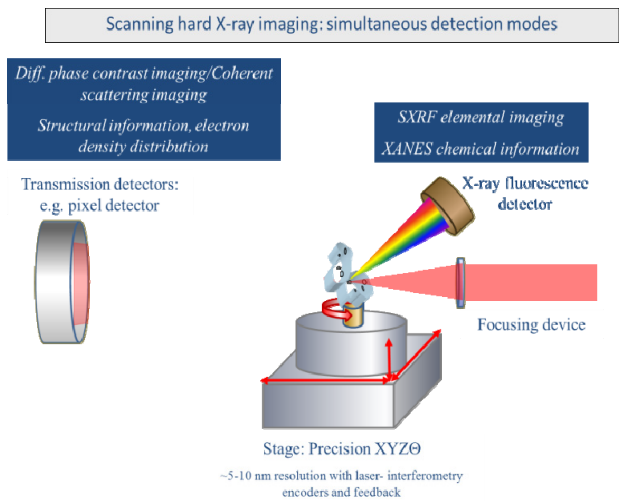


Figure 1: Schematic illustration of a multiple-detector fluorescence microspectroscopy experiment.

Multi-Detector Fast Acquisition System (FLYSCAN) Specifications

- Acquisition subsystems are independent and are synchronized by a common master clock trigger signal;
- Acquisitions must be done while systems are in motion to minimize data collection time;
- It must be possible to add a new acquisition subsystem to the experiment, without having to change existing software;

- Acquisition systems are running on various hosts (or crates), distributed on a TCP/IP network;
- All data collected on these various and independent subsystems must be recorded in a unique set of NeXus [2] files, encapsulating sufficient "metadata" (*e.g.,* experiment and sample identifiers, beamline parameters, time and date stamps) to simplify data reduction and analysis, facilitate reprocessing of the data in the future;
- Data organization within the files must be decoupled between acquisition and data processing to be able to add/remove detectors into the acquisition system without impacting data processing applications, even though the data organization within the files may be different.

SOFTWARE ARCHITECTURE

Data from each detector are collected by independent software applications (in our case Tango Devices), assuming all acquisitions are triggered by a unique master clock. Then, each software device streams its own data onto a common disk space, known as the spool (see Fig. 2). The data are stored in independent temporary NeXus files, with the help of a dedicated high-performance C++ library called NeXus4Tango. A dedicated asynchronous process, called the DataMerger, monitors the spool, and gathers all these temporary files into the final experiment NeXus file stored in SOLEIL common Storage System. Metadata describing the context of the data are also added in the final file by the DataRecorder device. All these actions are sequenced into a measurement process (Matlab, Python, Passerelle…)

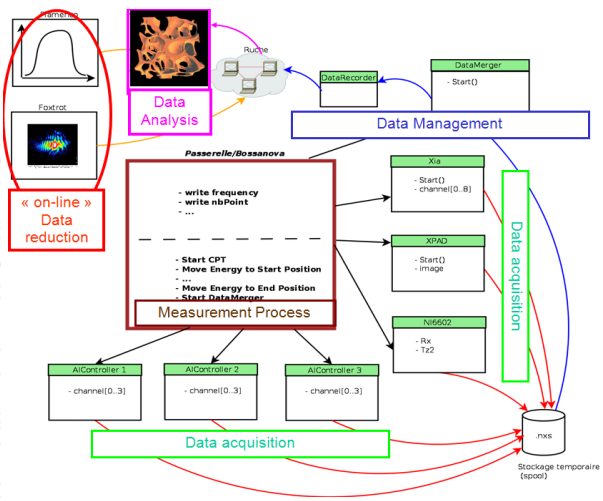


Figure 2: Data flow during the flyscan experiment. Multiple detector processes stream raw files to a temporary storage (spool), which are merged with metadata for permanent storage.

THE ACQUISITION COMPONENTS ON OUR NANOSCOPIUM PROTOTYPE

Acquisition subsystems used on the test bench for Nanoscopium are the following:

- An ADLINK DAQ-2005 analog-to-digital converter (ADC) board receiving signals from Si photodiodes measuring the X-ray beam intensity;
- A National Instrument (NI) PCI-6602 counter acquiring encoder positions of the 2 stages axis;
- A fast 2D photon-counting pixel array, XPAD [3], able to acquire up to 960 lines x 560 columns at 650 frames per second. (the XPAD is controlled with the LImA framework [4])

The master clock is another NI PCI-6602 configured as a trigger generator.

FIRST EXPERIMENTAL RESULTS

The FlyScan prototype has been deployed on the SOLEIL METROLOGIE beamline [5]. Official beamtime of few days has been dedicated to test in real condition this new acquisition system mode in the case of scanning differential phase contrast experiment [6].

In this experiment, a mapping (400 pixels x 40 lines) with a focused beam of a cross-nylon fiber was done in continuous mode. A master clock pulsed at 83 Hz (12 ms period), starts the acquisition of:

- The beam intensity (Fig. 3(a));
- The positions of the sample stages (Fig. 3(b));
- An image of the transmitted defocused beam (Fig. 3(c)) taken with one XPAD module (120 lines x 560 columns).

In order to construct differential phase contrast image, the center of gravity is calculated, on-line or off-line, on each XPAD image, along vertical and horizontal direction (Fig. 3(d) (e)).

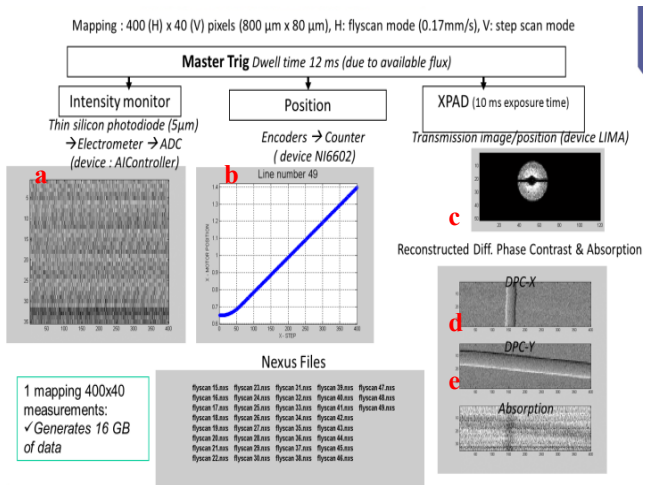


Figure 3: 1st results.

The architecture proved to be modular in terms of number and type of detectors. We have been able during the tests to easily integrate another 2D Detector (a Basler CCD camera).

DATA MANAGEMENT

Data Size and Network Throughput

There are two factors that have to be carefully taken into account: the volume of data and the speed of acquisition. A square mapping of 400 lines by 400 columns, with the same dwell time of 12 ms per point, generates 160 GB in 31 minutes (neglecting overheads incurred due to transferring the data over Ethernet and storing the files on hard disk). We estimate that once network and data storage performances are optimized (*using recent technologies like SSD disks and 10 Gbits Ethernet*), a single beamline such as Nanoscopium, working in flyscan mode with 2ms dwell time per point, could generate several hundred terabytes of data per year. This will present challenges to the data storage infrastructure and data processing capabilities.

Data Access

The software architecture proved to be modular and flexible at the Data Acquisition level. Nevertheless, the internal organisation of data with NeXus files changes depending on the number and types of detectors integrated in the data acquisition system. To hide these data organisation details to data analysis applications, SOLEIL develops with ANSTO the CommonDataModelAccess library [7].

CDMA is made of a core API that access data through a data format plugins mechanism and scientific applications definitions (i.e. sets of logically organized keywords defined by scientists for each experimental technique). Using an innovative "mapping" system between applications definitions and physical data organizations, the CDMA allows to develop data reduction applications regardless data files formats AND organizations.

Online Data Reduction

Because of the very large amount of data, it became evident already during the first experiments that data reduction applications should provide on-line data quality indicators to ensure whether the experiment is running as expected. The java COMETE [8] framework developed at SOLEIL allows us to reduce in "real-time" the large stack of raw images and spectrums to a small subset of reduced data, like for example reducing 160,000 XPAD images to the four images presented in this paper.

CONCLUSION/PERSPECTIVE

The following technical objectives are planned to be fulfilled before the end of 2011:

- stabilize all software components at the acquisition system level *(XPAD detector library , Tango device, Nexus4Tango library)*
- add the fluorescence XIA detector
- enrich existing COMETE data reduction applications
- integrate the CDMA C++ port within MATLAB and python to start integration in scientific data analysis applications

After the realisation, stabilisation, and test of the Nanoscopium prototype, an important project management phase will be to deploy FlyScans measurements on SOLEIL beamlines on a large scale basis, taking into account the impact on our data storage infrastructure and processing capabilities *(cluster and GPU)*.

We believe that this project is an important step for SOLEIL to prepare our computing resources to the scientific data deluge announced since years, but which is now becoming a reality.

REFERENCES

[1] TANGO framework: http://www.tango-controls.org/.
[2] NeXus file format : http://www.nexusformat.org.
[3] K. Medjoubi et al., Performances and applications of the CdTe- and Si-XPAD3 photon counting 2D detector, JINST 6 (2011) C01080.
[4] LiMA : https://forge.epn-campus.eu/projects/show/lima.
[5] M. Idir, P. Mercère, T. Moreno, A. Delmotte, P. Da Silva, and M. H. Modi, Metrology and Tests beamline at SOLEIL Design and first results. *AIP Conf. Proc.* 1234, *485-488* (2010).
[6] A. Menzel, C. M. Kewish, P. Kraft, B. Henrich, K. Jefimovs, J. Vila-Comamala, C. David, M. Dierolf, P. Thibault, F. Pfeiffer and O. Bunk, "Scanning transmission X-ray microscopy with a fast framing pixel detector," Ultramicroscopy, 110(9), 1143-1147 (2010).
[7] CommonDataModelAccess: ICALEPCS 2011 THCHAUST03.
[8] COMETE: http://comete.sourceforge.net.

EXPERIENCES IN MESSAGING MIDDLEWARE FOR HIGH-LEVEL CONTROL APPLICATIONS*

Nanbor Wang†, Svetlana Shasharina, James Matykiewicz, and Rooparani Pundaleeka
Tech-X Corporation, Boulder, CO 80303, U.S.A.

Abstract

Existing high-level applications in accelerator control and modeling systems leverage many different languages, tools and frameworks that do not interoperate with one another. As a result, the accelerator control community is moving toward the proven Service-Oriented Architecture (SOA) approach to address the interoperability challenges among heterogeneous high-level application modules. Such SOA approach enables developers to package various control subsystems and activities into "services" with well-defined "interfaces" and make leveraging heterogeneous high-level applications via flexible composition possible. Examples of such applications include presentation panel clients based on Control System Studio (CSS) and middle-layer applications such as model/data servers.

This paper presents our experiences in developing a demonstrative high-level application environment using emerging messaging middleware standards. In particular, we utilize new features in EPICS v4 and other emerging standards such as Data Distribution Service (DDS) and Extensible Type Interface by the Object Management Group. We first briefly review examples we developed previously. We then present our current effort in integrating DDS into such a SOA environment for control systems. Specifically, we illustrate how we are integrating DDS into CSS and develop a Python DDS mapping.

BACKGROUND AND INTRODUCTION

Accelerator control systems (ACS) coordinate the interactions among control hardware, data acquisition instruments, logging and data storage devices, and operator's interface. High-level accelerator control applications encompass activities such as operator control panels, tune measurement, orbit control, parameter save/restore, feedback, optic optimization, and parameter scanning that allow physicists and operators to control and reason accelerator behaviors in physically meaningful abstractions. There exist many tools and frameworks to help bring modern software engineering practices to the development and integration of lower-level hard real-time controls and the high-level soft real-time applications with great success.

Emerging Trends and Challenges

Control systems are often built on top of a set of existing tools and platforms that suit the needs of their target platforms. For examples, the EPICS [1] toolkit provides a standard for low-level controller architecture and a set of interoperable tools and engineering applications to assist control system developments. Depending on the scale of the target accelerator, high-level applications are often developed as a monolithic Graphical User Interface (GUI), a simple script, or a library routine. As in the case of generalized ACS control environments, many tools and frameworks such as Unified Accelerator Language (UAL) and Matlab Middle Layer Toolkit (MMLT), are available to assist the integration and interaction among high-level applications and device controllers (built, *e.g.*, using EPICS.)

All the different development environments and tools do not generally interoperate with one another. This is not a major issue for small- or medium-sized accelerators. However, such *ad hoc* approach no longer scales for modern large-scale accelerator facility such as the new NSLS-II, Project X, and the Intensity Frontier. Several Design Reports have called for a separate "service tier/middle layer" to provide devices and functional abstractions as units of integration.

SOA: Service-Oriented Architecture

SOA [2,3] has gained widespread acceptance in the business/enterprise software world as it has shown to facilitate the integration and composition of disparate software services across enterprises and businesses boundaries. Applying SOA principles in ACS is a promising approach in isolating and managing the complexity. In fact, many existing accelerator control systems have already adopted many SOA guidelines and principles. To address the needs and challenges of next-generation, large-scale accelerator control systems, we are enhancing the SOA environment for next-generation large-scale accelerator control systems to manage the complexity and contain the cost of developing future accelerator control systems and upgrading existing ones.

Message-Oriented Middleware

Traditionally, a SOA is often constructed on top of point-to-point request-reply, client-server communication middleware technologies. Middleware serves as the standard communication bus among different service. Examples of well-established SOA middleware include SOAP-based and RESTful Web Services, CORBA and Java RMI. However, there are still limitations with these point-to-point, request-reply, RPC-styled communication model as they impose scalability issues as the size and complexity of accelerator control systems grow.

The emerging DDS [4] is a new class of Message-Oriented Middleware (MOM) standard specified by the

*Work partially supported by US Department of Energy under contracts #DE-FG02-08ER85043 and #DE-SC-0000842, and Tech-X Corporation.
†nanbor@txcorp.com

Object Management Group (OMG). DDS complements RPC-styled client-server middleware and address their many limitations. DDS is a natural extension to many existing accelerator control frameworks. Furthermore, because DDS inherently supports many quality-of-service (QoS) policies such as message priority, rate, reliability, and deadline, that are necessary for mission-critical applications, DDS is a natural selection to act as the alternative service-bus in a SOA for control systems.

Figure 1 illustrates a SOA for high-level accelerator environment where applications exchange data using DDS as the common standard service bus. As shown in the figure, client applications can readily act as gateways to another enterprise service bus such as Web Services.

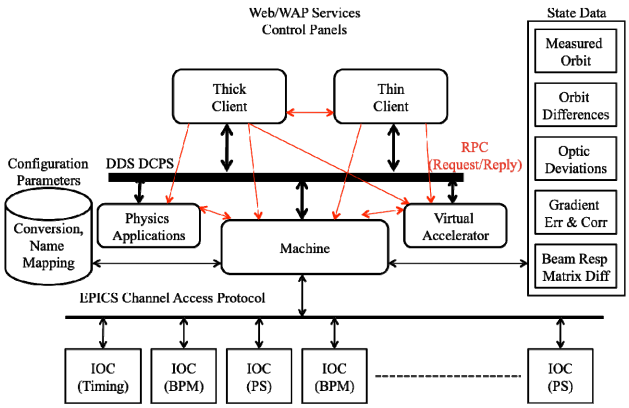


Figure 1: SOA for high-level applications over DDS.

DESIGNING DDS APPLICATION THROUGH PERFORMANCE TESTING

Because there are many ways to configure a DDS system, to better emulate various operation scenarios easily, we developed a DDS performance test suite and performed benchmarking tests with it. With the test framework, users will be able to develop various DDS runtime scenarios and experiment with various QoS policy combinations and evaluate their effect to the overall system performance. We design the test suite to be portable so that ACS developers can evaluate different DDS implementations easily.

Our performance test suite is built on top of the generic benchmarking application similar to the open-source Touchstone performance tool. Touchstone's benchmarking application provide the mechanism to instantiate test components such as transceiver and transponder for latency test, via special DDS messages for a set of control topics. Users can design and instantiate tests of different scales and with different combinations of QoS policies easily. Such approach allows users to create scenario-based performance evaluations easily.

Accessing the performance of messaging systems and overall applications provide critical information to help making key design decisions such as:

- Selection of DDS implementations: Different DDS implementations make different tradeoffs and adopt different implementation strategies to realize the movement of data from publishers to subscribers according to the QoS policies specified by all the entities involved. Some implementations also make such strategies configurable. Therefore, certain implementations or configurations perform better under certain operation environment while others ay scale better.
- Assisting in DDS configuration: DDS supports a rich set of QoS policies. Configuring a system using different sets of policies can affect the overall performance in different ways. For example, setting priority on one data stream can affect the overall behaviors of other data streams. Being able to perform tests to observe system behaviors at similar scales can provide essential guidance on design strategies.

EXAMPLE MIDDLE-LAYER SERVERS

To demonstrate the three-tier high-level application architecture, we implemented a general-purpose web-based optimization service with the help from our collaborator at Brookhaven National Lab (BNL). The service allows users to perform Twiss calculation and lattice optimization over the web.

Figure 2 illustrates the overall architecture of such a general-purpose web-based optimization service. This simple service allows accelerator physicists to submit lattice and optimization files to the service and returns the visual results of twiss calculations. This example provides

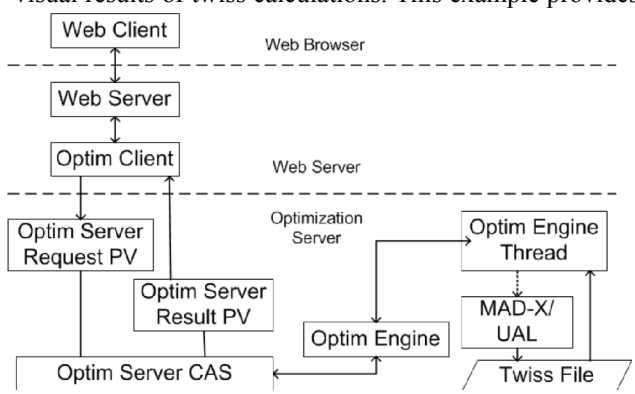


Figure 2: Example architecture of a middle-layer server.

an example "Software as a Service (SaaS)" prototype. It demonstrates the core idea on SOA using DDS and lays down the foundation for further, more complicated services. We implemented the prototype service using both MAD-X and UAL as the underlying compute engine. For messaging between the Web Server and the Optimization Server, we used both DDS and the experimental EPICS-DDS [5].

INTEGRATING DDS WITH THE CONTROL SYSTEM STUDIO

Fermilab's Intensity Frontier examplifies another usage scenario for a dual-message-bus SOA. There, they employ both DDS and EPICS in the control system. Subsystems that require real-time responses are built on

top of EPICS. Higher-level applications use DDS instead as the messaging mechanism.

There is a need to aggregate information and control various systems running on either EPICS or DDS together. Therefore, we are developing plug-ins for publishing and subscribing DDS topics from within the Control System Studio (CSS) [6]. The prototype plug-ins are modeled after org.csstudio.platform.libs.epics and its UI plug-ins. Another com.txcorp.soaac.css.dds.pv plug-in provides methods for publishing and subscribing topic-specific variable. Using our prototype, CSS widgets can subscribe or publish to variables as DDS topics using the familiar URI syntax such as "dds://topicname".

Although our prototype plug-ins demonstrate that DDS can be integrated into the CSS environment seamlessly, there are several limitations in our current design that we are working to improve:

1. Currently, we model the DDS topic structure closely after the EPICS PV data structure. We are working to relax this restriction so that a widget can subscribe to arbitrary variable within a DDS topic, e.g., "dds://device/part#control_point".
2. For a CCS application to subscribe to certain topics, their type-specific implementation must be generated and compiled into a .jar file, which then must be loaded into the application. We are working on adding support for a CSS application to read in topic structure and QoS definitions as XML files during runtime. This enhancement will eliminate the needs to compile and load application specific code into the CSS library.

SUPPORT PYTHON-DDS MAPPING (PYDDS)

Python language is a very versatile dynamic, object-oriented language that has gained great popularity among sceintists. Many scientists have expressed interests in interacting with DDS-based systems directly from within their Python codes. We have previously experimented with using DDS in Python. As illustrated in Fig. 3, our previous implementation of Python DDS support wraps all the topic-specific C/C++ codes generated by DDS tools into Python using SWIG or Boost.Python.

Although this approach serves the purpose, it is not compatible with Python's dynamic programming style. In particular, extra-steps outside of Python are required to generate the topic-specifc python mapping. Furthermore, when topic structures change, these wrappers have to be regenerated and repackaged in order for the actual Python application to use them. All these limitations interrupt the natural workflow of programmers and are not compatible the the dynamic language nature of Python.

In order to address these limitations, we are developing a new Python DDS (pyDDS) implementation. As shown in Fig. 4, we have moved the generation of topic-specific codes into Python. These topic-specific codes in-turn interace with generic DDS services. Using this approach, applications can source in topic structure definitions at

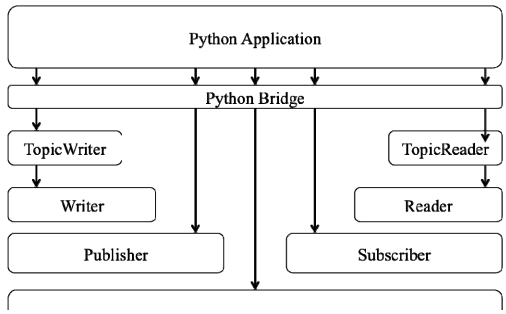


Figure 3: A straightforward Python DDS wraps generated topic-specific codes.

run time and generate the topic-specific readers and writers on the fly as pure Python classes. By eliminating the need to generate and load topic-specific wrappings in separate steps, the new pyDDS library is more compatible with dynamic programming nature of Python and also more easily accepted by Python developers.

The following listings demonstrate how to interact with a DDS data topic from within Python:

```
# First thing to do to use pydds
import pydds;
```

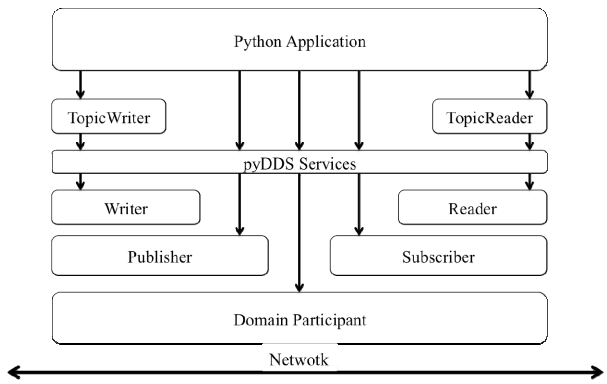


Figure 4: The new pyDDS implementation takes advantage of Python's dynamic language features.

To Join a Data Domain:

```
# Uers defines the dataspace runtime
# and pass it in to various other
# operations that need it.
 myDataspace =
   pydds.connect_dataspace
    ("Domain name", "Partition name")
```

To Manipulate QoS Policies:

```
myQoS = pydds.create_qos()
myQoS.set_reliable (3000000)
myQoS.set_transient()
myQoS.set_keep_last (3)
```

To Create Topic Reders/Writers:

```
# Creating/Finding a topic in global
# data space. Last argument specifies
# the URI to the topic definitions
helloTopic = pydds.createTopic
 ("TopicName",
  myDataspace,
  myQoS,
  file:///HelloWorld.idl#HelloTopic)

# Now create reader/writer objects
helloReader =
 helloTopic.create_reader(readerQoS)
helloWriter =
 helloTopic.create_writer(writerQoS)
```

To Write and Read Data Samples:

```
# Creating a sample
helloSample =
 helloTopic.create_sample
  (message = "John Smith",
   repeat = 3)

# Publishing the sample
status =
 helloWriter.write (helloSample)

# Simple read/take
[samples, infos] = helloReader.read()
sys.stdout write(samples[0].message)
```

We plan to model listener-based callback data read interface after the Twist or Trellis libraries. Furthermore, we would also like to take advantage of the emerging eXtensible Type Standard which is being ratified by the OMG [7,8].

CONCLUDING REMARKS

This paper describes how a dual-service-bus SOA provides an ideal development environment that promotes modulization of high-level application components and facilitates dynamic composition of high-level applications. In the context of the high-level application environment, it means flexibility in selecting and connecting the most appropriate modeling algorithms and programs. To support such use case, we are bringing the next generation data-centric publish-subscribe middleware called DDS to this environment by investigating various usage scenarios and programming patterns of DDS in ACS. Furthermore, we are developing tools and libraries to enable ACS developers to leverage the new technologies. We developed a DDS performance test suite and used it to benchmark and evaluate various open source and commercial DDS implementations. We also implemented a set of example Web-based high-level application on top of open source DDS implementations.

Other than demonstrating how a ACS can utilizes DDS middleware technologies, we are also developing tools to help facilitate the adoption of DDS. We implemented a set of prototype plug-ins for Control System Studio to monitor and emit DDS data stream. We also developed a prototype Python mapping for DDS to enable Python applications to participating in DDS data exchange. We are in the process of hardening these prototypes and enhance their robustness and usability.

ACKNOWLEDGMENTS

The authors would like to thank Nikolay Malitsky of Brookhaven National Lab for his help in formulating the middle-layer server examples. Similarly, we wish to thank Jim Kolwalkoski, Marc Paterno, Kurt Biery, and Erik Gottschalk of Fermilab for discussing the needs and requirements of their projects.

REFERENCES

[1] L. Dalesio *et al.*, "The Experimental Physics and Industrial Control System Architecture," ICALEPCS'93, Berlin, Germany, October 1993, http://www.aps.anl.gov/epics/

[2] Dirk Krafzig and Karl Banke and Dirk Slama. Enterprise SOA – Service-Oriented Architecture Best Practices. Prentice Hall, 2005.

[3] Eric Newcomer and Greg Lomow. "Understanding SOA with Web Services," Addison Wesley, New Jersey, 2005.

[4] OMG, "Data Distribution Service for Real-time Systems, Version 1.2," formal/07-01-01, http://www.omg.org/cgi-bin/doc?formal/07-01-01

[5] N. Malitsky *et. al.*, "Prototype of a DDS-based High-Level Accelerator Application Environment," ICALPCS09, Kobe, Japan, Oct 2009.

[6] Desy, SNS, BNL, "Control System Studio," http:// http://css.desy.de/content/index_eng.html

[7] OMG, "Extensible and Dynamic Topic Types for DDS," http://www.omg.org/spec/DDS-XTypes

[8] N. Malitsky *et. al.*, "DDS XType-based Machine Server," ICALEPCS'11, Grenoble, France, Oct. 2011.

RUNNING A RELIABLE MESSAGING INFRASTRUCTURE FOR CERN'S CONTROL SYSTEM

F. Ehm, CERN, Geneva, Switzerland

Abstract

The current middleware for CERN's Controls System is based on two implementations: CORBA-based *Controls MiddleWare* (CMW) and *Java Messaging Service* (JMS). The JMS service is realized using the open source messaging product *ActiveMQ* and had became an increasing vital part of beam operations as data need to be transported reliably for various areas such as the beam protection system, post mortem analysis, beam commissioning or the alarm system. The current JMS service is made of 18 brokers running either in clusters or as single nodes. The main service is deployed as a two node cluster providing failover and load balancing capabilities for high availability. Non-critical applications running on virtual machines or desktop machines read data via a third broker to decouple the load from the operational main cluster. This scenario has been introduced last year and the statistics showed an uptime of 99.998% and an average data serving rate of 1.6GByte per minute represented by around 150 messages per second.

Deploying, running, maintaining and protecting such messaging infrastructure is not trivial and includes setting up of careful monitoring and failure pre-recognition. Naturally, lessons have been learnt and their outcome is very important for the current and future operation of such service.

THE CERN CONTROL SYSTEM

Today, the CERN control system is constructed as a three-tier architecture with real-time processes reading signals from the equipment, data processing services and graphical interfaces to display the data.

The signals produced by the accelerator equipment are read out by real-time C++ processes running on around 1600 so-called *Front End Computers* (FEC) and published further to higher level services where it is filtered, evaluated or correlated with other data. The elements in the last and highest level of this architecture are *Graphical User Interfaces* (GUI) mostly written in Java which display the data or let operators in the *CERN Control Centre* (CCC) take direct control on certain equipment by enabling them to set hardware parameters.

All involved components communicate via the Controls MiddleWare (CMW) to publish and/or exchange data. It is actually composed of two products: an in-house developed CORBA [1] based solution (RDA) and ActiveMQ [2], an open source implementation of the *Java Messaging Service* (JMS) [3] API.

RDA is implemented in C++ and Java and provides the point-to-point communication between the involved peers using a physical connection. This allows low latency with a high grade of control. The network and CPU load caused by distributing the data to the recipients resides on the publishing instance.

ActiveMQ is written in Java and was released the first time in 2004 and is in use at CERN for Beam Controls since 2005. It implements the JMS 1.2 specification and fulfils the *Message Oriented Middleware* paradigm where the communication between sender (*producer*) and the recipient (*consumer*) is relaxed by the introduction of an intermediary component (*messaging broker*). By using the JMS API application modules can be distributed over heterogeneous platforms and thus reduces the complexity of developing applications that span multiple operating systems and communication protocols. Producers publish data to the brokers and consumers register their interest in specific messages via the Publish-Subscribe (*Topic*) or Point-to-Point (*Queue*) mechanism. It is then the broker's responsibility to distribute the data reliably.

As a result of this intermediary broker and due to the additional network hop the communication latency increases. However, for the involved systems of the CERN Controls System this latency is acceptable.

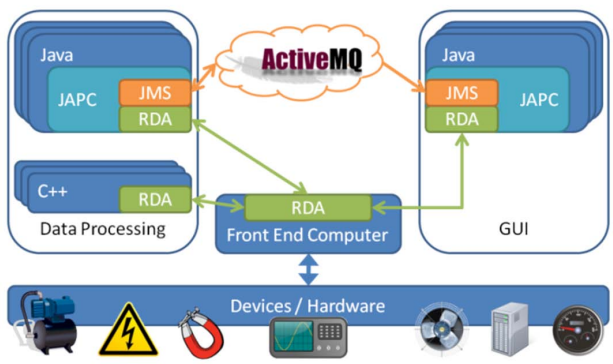


Figure 1: Usage of the Control System Middleware.

As shown in Fig. 1 the ActiveMQ infrastructure is deployed for service to service and service to GUI communication. CMW enables direct communication such as sending commands, tuning hardware equipment and monitoring their parameters. Although not required, data processing services using JMS are exclusively implemented in Java and services which use RDA may be C++ processes as well. In both cases the *Java API for Parameter Control* (JAPC) is used to abstract the underlying middleware.

Choice of a Centralized Messaging System

The main reason for having a messaging system such as ActiveMQ acting as a relay is the clear advantage of outsourcing the work of data distribution to a dedicated entity which is actually made for such a purpose. It originates from the stock market system where the same technology is used to serve thousands of consumers. The control system at CERN does not reach this number but shows very similar requirements in terms of flexibility, scalability and robustness.

CURRENT DEPLOYMENT

The current deployment of ActiveMQ installations consists of 14 brokers used for production and 4 for development. Depending on a project's needs either a single broker or a *broker cluster* for failover and load balancing purposes is set up. A cluster consists of usually two or more interconnected brokers which exchange information about producers and subscribers to forward a message if required. In case one of the brokers crashes clients automatically reconnect to another one in the same cluster.

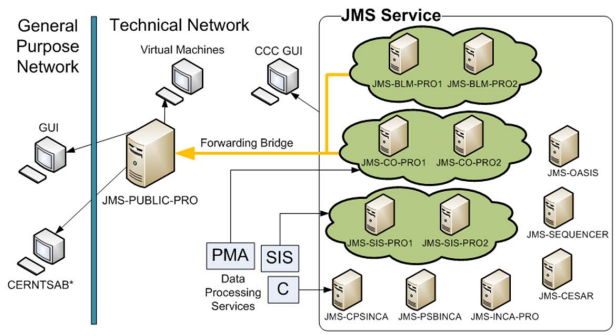


Figure 2: Overview of the Messaging Service.

As Fig. 2 illustrates there are currently 3 main clusters installed for selected middleware services such as the *Software Interlock System* (SIS) and the *Beam Loss Monitors* (BLM) system. The third one is called JMS-CO-PRO and is shared among projects. Alternatively, a project dedicated broker is deployed on the same machine as the related middle tier server. Because these are deployed as single instances it is irrelevant to have a redundant broker service (and hence a higher management load). Assigning

As Fig. 2 also shows that a *data forwarding bridge* to a broker for non-operational clients has been deployed. These clients are GUIs which are not used for beam operations but by developers or experts or as information displays running on windows terminal servers or on virtual machines. Data is forwarded from the main JMS Service to this public read out broker which again redistributes the data to the subscribers. This setup provides several advantages:

- First, a separation of critical clients from non-critical ones allows reducing the additional load caused by latter noticeably.
- Secondly, it enables reading data from outside the closed technical network without exposing the main service machines but a single broker.
- And thirdly, it gives the possibility to instantly remove load from the main service by shutting down the public read broker.

MONITORING

Like other services a messaging broker system needs to be carefully monitored to proactively identify upraising problems and to react to them accurately and immediately. Not only default machine metrics such as network, CPU and disk usage take part in this monitoring activity but additionally specific tools have been developed to collect more – broker specific - information.

Instrument JMX and Sending of a Test Message

One vital prerequisite for such tools is to be able to get an insight view into a running broker. ActiveMQ provides this functionality via the *Java Management Extensions* (JMX) [4] interface and it is easily integrated with existing diagnostic and monitoring tools. Only a subset of the exposed metrics is actually chosen to evaluate the broker health periodically. Information such as memory and storage usage as well as the processed messages since the last iteration is available. However, these numbers may not be sufficient to make a complete statement on the broker condition. Therefore, the *message processing speed* (mps) is measured every 5 minutes to evaluate the time a message takes to go through a broker or a full broker cluster. This is done by an external monitoring agent which sends and reads a test message and compares the submission and receive timestamp. Analogue to the connected applications a high latency is detected very quickly and subsequently treated as a potential problem.

Thresholds are used to determine the severity of the result to then inform service managers via eMail and the *Short Message Service* (SMS). As an example: an mps of 100ms would correspond to a warning, 1000ms to an error.

Topic Monitoring Tool

An in-house developed *Topic Monitoring Tool* (TMT) listens actively to all messages and stores statistics like throughput and size using the *Round Robin Tool* [5] database technology. This again can be used to visualize a history view on the average values like MRTG [6]. An example of the usage of this view is shown in Fig. 3. A producer sending at irregular high data rate was recognized (left red area) and a restart subsequently solved the problem (green area).

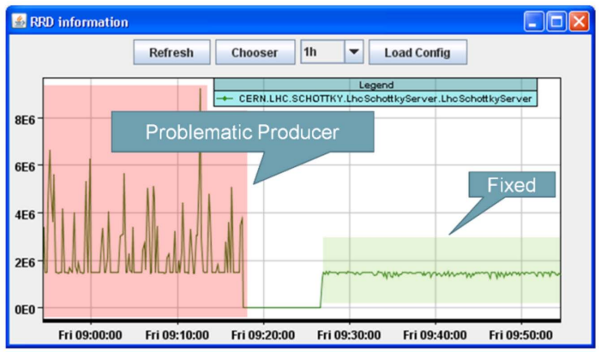


Figure 3: Detecting a producer problem using the TMT.

PERFORMANCE AND RELIABILITY

When it comes to performance and reliability ActiveMQ has proven to be very stable in the present environment and configuration. In 2010 the JMS-CO-PRO and BLM services handled together up to 2.57TByte per day in data volume and reached a service availability of 99.998%. This includes downtimes due to kernel and machine upgrades. Single broker services showed a lower availability in average because of missing redundancy. But 210 consecutive days of uptime with an average of 120 messages per second as one example shows that a single broker instance is very reliable.

Another example for load balancing is the BLM system: it sends out messages of 2 MByte which are read by a large set of consumers. While investigating delay of messages it turned out that the network card in the broker machine was actually a bottleneck. As a consequence, a second broker on a separate machine was deployed to balance the network load. Alternatively, binding the second broker to a second installed network card in the same machine would have had the same effect.

MESSAGE USAGE PATTERNS AND SERVICE LEVEL AGREEMENTS

The usage pattern varies from rather small messages at high frequency (1KByte payload at 100Hz) to large messages at low rate (2MByte payload at 0.5Hz). In case of latter for example, around 25 clients constantly request this data. Other services such as the *Post Morten Analysis* (PMA) system show a very different usage scenario: they send high bursts of small messages only after a beam dump.

Depending on the characteristics of the published data the number of reading clients may vary. In theory, all CCC consoles may read all data at the same time. However, in practice this is not the case. Due to the organization of the CCC an average of 20 GUIs is taken for the data distribution load. This means one message has to be delivered to 20 consumers.

Service Level Agreements

In order to protect a customized messaging service for a project *Service Level Agreements* (SLA) had been established. They describe on a high level the project's needs and message usage patterns and thus help to decide how to realize these requirements. Both sides then have to agree on the SLA and make sure that the rules are respected.

For example: the BLM SLA states that the messaging service must sustain a constant rate of 2MByte payload at 1Hz with 20-30 clients. If much more data is sent than agreed then there is no guarantee that the service will remain stable.

These SLAs also allow tracking of changes of the project's requirements and ease setting up monitoring thresholds. They are based on the experience from the *Large Hadron Collider* (LHC) startup in March 2010 where the amount of data which was estimated was exceeded by a factor of 2.5 (see Fig. 4). A possible messaging service failure was prevented by a precautious upgrade of the machines. This case shows that the prediction for load in such an environment with a great variety of applications (and developers) is not always definitive and final.

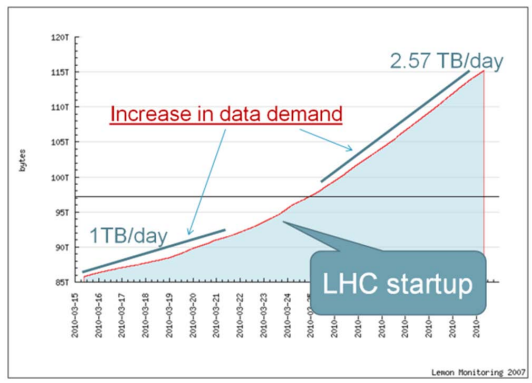


Figure 4: Data demand during start up of the LHC.

PROBLEM ANALYSIS IN PRODUCTION

When having a system which is vital for (beam) operations it is important to be able to quickly analyse a problem to reduce costly downtime. In contrary to a point-to-point communication like CMW a centralized messaging system always faces the problem of being a potential single bottleneck. However, there is also good side. Because of this single point of failure the problem analysis is limited to a fewer places.

For ActiveMQ there are the following instruments for such investigations:

- JMX console showing very detailed information like connections, subscriptions, etc.
- Test clients for reading data from topics
- Changing broker log level without a restart
- Dump of available data via JMX to a SQLite [7] database for easier extraction

The experience shows that most problems caused by an ActiveMQ broker were based on changes which were introduced shortly before (e.g. upgrade or configuration changes).

There is also the much more dangerous possibility that an effect occurs days or weeks later. For example: a new

broker version was tested in an integration test bed before going to production. After the upgrade the service ran smoothly for the next weeks but failed at a certain point due to a bug in the configuration settings of the broker software. As a result, the JMS-CO-PRO cluster was not available anymore to the clients and many critical services stopped working.

Unfortunately, there is no straight solution which is feasible in time and effort for such situations. As for many other services it must be noticed that such incidents may happen (although rarely) despite all monitoring and precautious actions which may reduce the probability but never eliminate such cases.

More important is the ability to know strategies and solutions to quickly resolve the situation.

SUMMARY

ActiveMQ was and is a good choice for decoupled messaging for the CERN's Control System and has proven to be very stable. In particular the strength of scaling linearly makes it inevitable for an environment where the number of reading applications is very dynamic and data demand is growing at the same time. Because unexpected high load is possible it is important to dimension machine resources sufficiently.

In this context, monitoring is a vital part of operation and the evaluation of the recordings must take influence on future deployment decisions. Effective service downtime is reduced by deploying a messaging service per usage domain or project.

It is highly recommended to set up SLAs to track the (growing) user needs. They help to adapt the system to usage scenarios before they are put into production and support setting up monitoring thresholds.

REFERENCES

[1] CORBA, Common Object Request Broker Architecture, OMG, http://www.omg.org
[2] Apache ActiveMQ: http://activemq.apache.org/
[3] JMS Java Messaging Service, http://www.oracle.com
[4] Java Management Extension, Oracle: http://www.oracle.com
[5] Round Robin Tool, Tobias Oetiker: http://oss.oetiker.ch/rrdtool/
[6] MRTG, The Multi Router Traffic Grapher, http://oss.oetiker.ch/mrtg/
[7] SQLite Database: http://www.sqlite.org/

A LEGO PARADIGM FOR VIRTUAL ACCELERATOR CONCEPT

S. Andrianov*, A. Ivanov†, Eu. Podzyvalov‡, SPbSU, Saint-Petersburg, Russia

Abstract

The paper considers basic features of a Virtual Accelerator concept based on LEGO paradigm. This concept involves three types of components: different mathematical models for accelerator design problems, integrated beam simulation packages (i. e. COSY, MAD, OptiM and others), and a special class of virtual feedback instruments similar to real control systems (EPICS). All of these components should interoperate for more complete analysis of control systems and increased fault tolerance. The Virtual Accelerator is an information and computing environment which provides a framework for analysis based on these components that can be combined in different ways. Corresponding distributed computing services establish interaction between mathematical models and low level control system. The general idea of the software implementation is based on the Service-Oriented Architecture (SOA) that allows using cloud computing technology and enables remote access to the information and computing resources. The Virtual Accelerator allows a designer to combine powerful instruments for modeling beam dynamics in a friendly way including both self-developed and well-known packages. In the scope of this concept the following is also proposed: the control system identification, analysis and result verification, visualization as well as virtual feedback for beam line operation. The architecture of the Virtual Accelerator system itself and results of beam dynamics studies are presented.

INTRODUCTION

In modern accelerator physics there gives much attention both to the information support problems as a process of constructing the accelerator systems and the maintenance of corresponding experiments. In other words, it is necessary to create an integrated package of special software oriented for modeling of operating characteristics of an accelerator considered as a control system. Here we mean a distributed control system for both electromagnetic structural elements and a beam state. In the corresponding package is it necessary to take into account consider both the most important procedures like measurement of the beam, control elements characteristics and technologies required for a comprehensive evaluation design for each of these key questions before manufacturing and testing physical prototypes. In the VA package there are integrated different approaches to both numerical modeling problems beam dynamics and beam control system operation. More over, the corresponding approaches involve necessity to carry out a global optimization procedure of an accelerator in whole (or its components) with a given criteria family.

The Virtual Accelerator package implements a unique hybrid approach to modeling the dynamics of the beam, based on a combination of a large number of proven methods of analysis based on a combination of numerical and symbolic operations, in which the model as components of the accelerator and beam models are combined with virtual models and synthesized control fields. By combining environmental test and virtual simulation the package VA speeds up the design process and makes it more accurate and stable because it has a built-in validation of models based on test results.

The VA package allows users to:

- combine proven components models with virtual models of new components;
- develop reliable data on the beam with a virtual simulation and, if possible, the results of environmental tests;
- improve the accuracy and speed of simulation at the system level, using real data and reliable virtual model;
- adapt the virtual assembly control system for assessing critical characteristics of the beam;
- estimate the characteristics of the whole control system before natural testing of a prototype.

Virtual Accelerator package provides users a set of well-proven packages of modeling of dynamics (such as MAD, Trace3D, COSY Infinity, MaryLie and others developed by the authors of this article, see, for example, [1]), as well as the software package that allows you to implement the concept of a global optimization (see, for example, [2]). In the process of the corresponding research the model includes the necessary edge effects, nonlinear aberrations of different nature, the effects of space charge and so on. For problems of long-term evolution there is supposed to use different methods of symplectic integration. This gives us the necessary accuracy for long time evolution (it is very important for cyclic accelerators).

The VA package will provide “transparent” access to data modeling and its results, making it an integral part of the process numerical simulation and monitoring of existing accelerator facilities. Construction and integration of computational models with specific characteristics is a key point of virtual simulation. Therefore, the package should

* sandrianov@yandex.ru
† 05x.andrey@gmail.com
‡ podzyvalov@mail.ru

contain necessary tools to edit the structure of the accelerator control system, models of control fields, the description of the beam as a family of particles (including taking into account its charge), modeling and assembly for fast virtual models with the necessary visualization tools. The VA package allows users to combine a numerical model of the system-level virtual and the resulting test model components. It should also be pointed out that this package provides several opportunities for optimization of one or more of the criteria (including antagonist). With the help of neural network modeling techniques designer can determine the possible structures corresponding to their requirements. Using effective methods and optimization algorithms (e. g. using genetic algorithms, see, for example, [2]) one can find a set of appropriate solutions. The global optimization concept is proposed to use the method of finding rational alternative, based on the combination of the decision and Pareto method optimization using both integral criterion and using different methods and algorithms for accelerator modeling. This technique was tested for the solution of optimization problems of control systems particle beams [2]. It is also expected to use the methods and algorithms involving expert systems for optimization of complex systems with incomplete information. This approach allows the concept to use data mining methods and technologies for control problems in complex systems. For the analysis of investigated alternatives are used both numerical modeling procedures and the corresponding analytical model using the tools and techniques of Computer Algebra [3].

It should be noted that all above mentioned approaches are based on the LEGO-objects concept (see, for example, [4, 5, 6]), which on one hand allows to unify the mathematical objects at all stages of modeling and maintenance of the necessary computational procedures and on the other – to provide the necessary flexibility and scaling the appropriate software.

SOME PRINCIPLES OF THE LEGO-OBJECTS CONCEPT

The main idea of the LEGO-objects concept is based on a deep hierarchy of *elementary blocks* for presentation of both *physical elements* and corresponding *computational tools*. These blocks can be separated in three groups: the first corresponds to physical objects, the second has a virtual character and the third describes computational procedures. For the second type a beam line researcher can not relate such virtual block to any physical element. As an example of such *virtual element* the fringe field description block can be mentioned. These elementary blocks are used for decomposition of the whole physical system into a set of subsystems. The symbolic representation of these blocks allows to create databases of elementary classes of such subsystems. Manipulation by these classes is similar to manipulation by LEGO-blocks. The similar approach is developed in some works [7], but the authors do not use symbolic representation for solution to be found.

The first two types of LEGO-objects can be connected with objects having some different (not only physical) interpretation. LEGO-objects of the third type are accountable for computational procedures of different kind. As an example we can mention two groups of methods of integration: non-symplectic and symplectic integration methods. So the VA package has to involve other types of LEGO-objects which generated by operations above LEGO-objects of the first two types. All types of LEGO-blocks have corresponding presentation on at all phases of simulation process: from physical models up to computing models. Including and excluding any elementary block have to be realized quietly, without distortion of the whole model. This approach is an essence of the *dynamic modeling paradigm* [1].

The above pointed approach is based on a hierarchical sequence of approximation models, which can be built for an investigated beam line. The modeling process (including optimization procedures) has several major phases:

- *the initial phase of problems formalization on physical understanding level — creation of a set physical models for problem under consideration;*
- *construction of approximating models in conformity with experiments data — creation of a hierarchy of approximating models;*
- *the phase of problems formalization on mathematical understanding level — creation of a hierarchy of mathematical models;*
- *creation of computing models (including algorithms and corresponding software) realized mathematical models;*
- *realization of the computing process for modeling and optimization — a set of computing experiments;*
- *the phase of the interpretation of computing results and testing on adequacy to the physical model.*

The basic stages of computational processes are presented in Fig. 1. All presented modules are not only necessary for correct realization of computational experiments, but they gives designer effective tools for necessary computational experiments. It is necessary to note that usage of symbolic algebra tools give us additionally the necessary flexibility and effectiveness of the computational process.

The process of ordering and accumulation of used knowledge, decomposition of complex systems into a set of simple subsystems (real and/or virtual) must guarantee good maintainability, reasonability, and extensibility of designed codes for numerical modeling. On this step a researcher forms his understanding of practical problems under study. Complexity of modern problems have to lead to necessity of using of databases and knowledge bases ideology and corresponding tools. On this phase of modeling it is necessary to pay attention to classification and

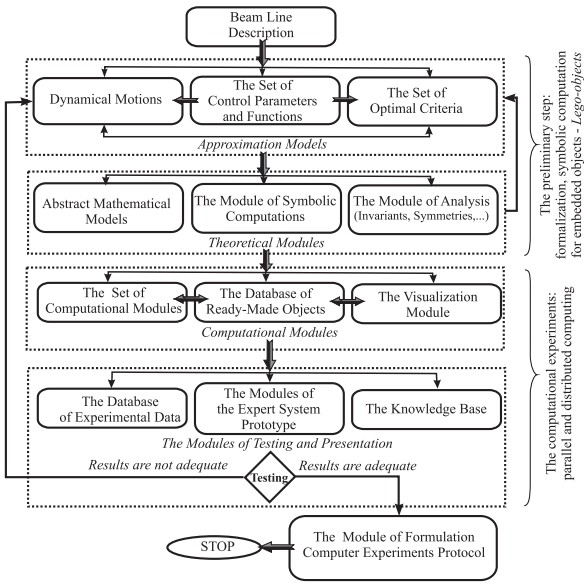


Figure 1: The total cycle of the computational experiment.

systematization of used knowledge about the problem under study (in term of LEGO-objects this means an in-depth classification and structuring). Similar approach may be suggested on the phase of mathematical models forming too. Moreover, the deep mathematical formalization helps researchers to create adequate mathematical models easily than it is done on the previous phases. Naturally effectiveness of computer algebra methods and codes usage depends on selected mathematical methods and tools. These methods must admit a deep hierarchy of LEGO-objects. These objects have their own characteristics needed for sewing together in a whole physical element.

All elementary blocks have their representatives on all phases of modeling process: from physical up to computing models. Including and exclusion either elementary block must be realized quietly, without distortion of the whole model. This approach is realized using dynamic modeling paradigm [8]. It should be note that the visualization procedures must be included to the computing model as its integral part. Indeed the results of visualization process help us understand dynamical processes occurred in either situation. Here it is necessary to tell some words about optimization procedure particularly. Usage of symbolic representation for problems under study can realize the corresponding optimization process more effectively (see, for example, [2]). As the conclusion it is necessary to note that realization of discussed approach allows to carry out the analysis in a special virtual laboratory – Virtual Accelerator, that is not only favorable from the economic point of view, but also allows us to investigate many effects more detail and carefully, usually this is unacceptable to only natural experiments.

CONCLUSION

The above described approach can be realized using two mathematical and computational frameworks. The first is enough traditional for the most published Virtual Accelerator concept: a combination well known and approved beam line packages and EPICS instruments. The second is based on the first approach (based on numerical modeling first of all) and more extended application of symbolic computational procedures. In our case we use the matrix formalism for Lie algebraic tools [9]. This approach gives the designer more effective and extensible toolbox. The most of modules presented on the Fig. 1 were realized using this formalism and were applied for different problems of beam physics (see, for example, [1, 10, 11]).

REFERENCES

[1] S.N. Andrianov, “Dynamical Modeling of Control Systems for Particle Beams”, SPbSU, Saint Petersburg, 2004 (in Russian).

[2] S.N. Andrianov, N.S. Edamenko, E.A. Podzyvalov, “Some Problems of Global Optimization for Beam Lines”, Proc. of 4-th Intern. Conf. on Physics and Control (PhysCon 2009), Catania, Italy, 2009, http://lib.physcon.ru/download/p1998.pdf.

[3] S.N. Andrianov, “A Role of Symbolic Computations in Beam Physics”, COMPUTER ALGEBRA IN SCIENTIFIC COMPUTING. Lect. Notes in Comp. Scie., 2010, Vol. 6244/2010. p. 19.

[4] S.N. Andrianov, “Lego-Technology Approach for Beam Line Design”. EPAC 2002, Paris, France, 3–7 June, 2002. p. 1091.

[5] Andrianov S.N., “A Modular Approach for Beam Lines Design”, COMPUTER ALGEBRA IN SCIENTIFIC COMPUTING. Lect. Notes in Comp. Scie., 2011, Vol. 6885/2011. p. 37.

[6] Andrianov S.N., “Some Problems of Optimization Procedure for Beam Lines”, EPAC 1998, Stockholm, Sweden, June 22–26, 1998. p. 1091.

[7] Cai,Y., Donald, M., Irwin, J., Yan, J.: “LEGO: A Modular Accelerator Design Code”. SLAC-PUB-7642, August 1997.

[8] Andrianov S.N., “Dynamic Modeling Paradigm and Computer Algebra”, Proc. of the 9-th Intern. Conf. Computational Modelling and Computing in Physics, Dubna, Russia, Sept. 16–21, 1997, p. 60.

[9] Andrianov S.N., “A Matrix Representation of the Lie Algebraic Methods for Design of Nonlinear Beam Lines”, AIP Conf. Proc. Vol. 391 (Eds. J.J. Bisognano, A.A. Mondelli), N.Y., USA. 1996. p. 355.

[10] Andrianov S.N., Edamenko N.S., Tereshonkov Yu. “Some Problems of Nanoprobe Modeling”. Intern. J. of Modern Physics A, 2009, V. 24, No 5. p. 799.

[11] Andrianov S.N., Edamenko N.S., Dyatlov A.A. “Algebraic Modeling and Parallel Computing”. Nuclear Instruments and Methods. Ser. A Vol. 558, 2006, P. 150.

EUROPEAN XFEL PHASE SHIFTER: PC-BASED CONTROL SYSTEM

E. Molina Marinas*, J.M. Cela-Ruiz, J. de la Gama, A. Guirao, L.M. Martinez Fresno, I. Moya, A.L. Pardillo, S. Sanz, C. Vazquez, CIEMAT, Madrid, Spain

Abstract

The Accelerator Technology Unit at CIEMAT is in charge of part of the Spanish contribution to the European X-Ray Free-Electron Laser (EXFEL) [1]. This paper presents the control system of the Phase Shifter (PS), a beam phase corrector magnet that will be installed in the intersections of the SASE undulator system. Beckhoff has been chosen by EXFEL as its main supplier for the industrial control systems. Beckhoff Twincat PLC architecture is a PC-based control technology built over EtherCAT, a real-time Ethernet fieldbus. The PS is operated with a stepper motor, its position is monitored by an incremental encoder, and it is controlled by a Twincat-PLC program using the TcMC2 library, an implementation of the PLCopen Motion Control specification. A GUI has been developed in LabVIEW instead of using Beckhoff visualization tool. The control system for the first and second prototype devices has been developed in-house using COTS hardware and software. The specifications request a repeatability of ($\pm 50\mu$m) in bidirectional movements and ($\pm 10\mu$m) in unidirectional movements. The second prototype can reach speeds up to 15 mm/s.

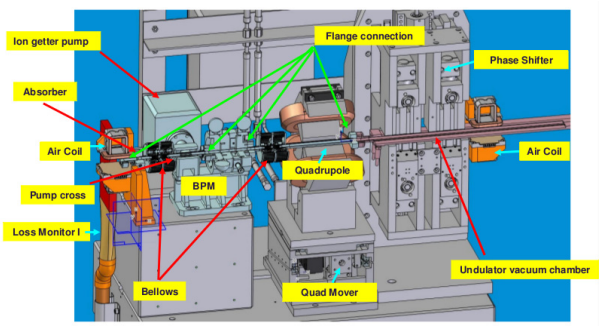


Figure 1: EXFEL Undulator System intersection.

INTRODUCTION

The undulator systems of the European XFEL have a total approximate length of 200 m. They are divided into cells which comprise a 5m long undulator segment and a 1.1 m long intersection, Fig. 1, where several devices needed to adjust the beam properties are located [2]. 92 sets of components for the intersections will be supplied by CIEMAT, as part of the spanish in-kind contribution to the European XFEL. The control system is also a part of this contribution.

* eduardo.molina@ciemat.es

One of the components of the intersection is the Quadrupole Mover (MV) [3], a biaxial (X-Z) moving table that aligns a 70 kg quadrupole magnet within ± 1.5mm in two dimensions with repeatability better than $\pm 1\mu$m. Two 5-phase stepper motors move the table in the horizontal and vertical directions. Positions are monitored with two LVDTs.

Another device is the Phase Shifter [4], see Fig. 2. Its function is to correct the phase of the electron beam with respect to that of the radiation field when the wavelength is changed. This is achieved by opening or closing the gap between two sets of magnetic modules geared by a double left-right hand threaded lead screw, which, in turn, is actuated by a stepper motor fitted with a 40:1 gearbox [5].

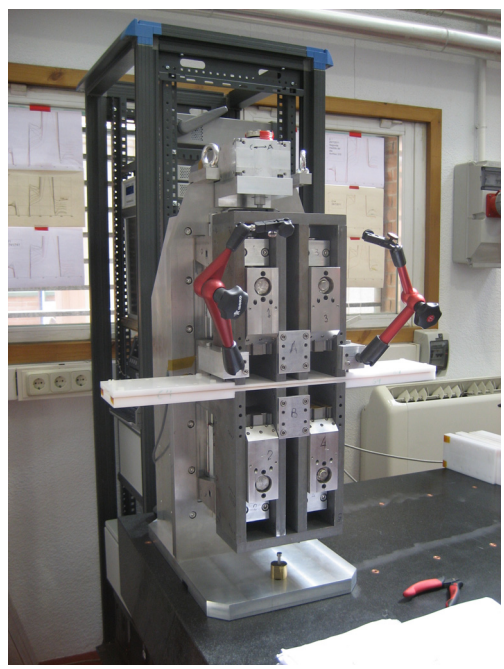


Figure 2: Phase Shifter.

The Intersection Control Rack (ICR) will house an EtherCAT bus coupler plus all the modules needed for operation of the PS and the Quadrupole Mover. Electrical protection modules and low-noise power supplies are to be mounted in order to improve its long term reliability. The bus coupler links the ICR to the Undulator Rack, where an embedded PC will control the corresponding undulator cell.

For protoype testing (MV, PS), dedicated control cabinets have been assembled with the required Beckhoff control modules (bus coupler and I/Os), along with motor drivers, power supplies and signal conditioning units. The

PLC control unit, for development purposes, is always a standard PC running Beckhoff Twincat PLC under Windows, with an extra Ethernet PCI card connected with a standard cat-5 cable to the EtherCAT bus coupler in the cabinet. To expand the possibilities of the native Beckhoff SCADA interface, a GUI has been written in LabView to operate the PS from the PC. This is part of CIEMAT's effort to produce a software which can be used for both testing and validation of the prototypes and the production batch.

In this paper, the components of the PS control system are presented in detail.

PHASE SHIFTER CONTROL SYSTEM

PC-based Automation

Many flavours of PC-based automation solutions are being offered nowadays by all major control technology companies. Due to increasing performance and decreasing cost of hardware components, embedded PCs equipped with flash memory cards or solid-state drives can be used in an industrial environment replacing the traditional PLCs.

Following hardware progression, many software packages populate the market, turning any PC into a real-time controller, and thus allowing programming, diagnostics and configuration of devices connected either to the PC directly or through fieldbus cards, while ensuring deterministic real-time behavior. There is also a growing presence in the automation industry of real-time communication protocols based on Ethernet which are gaining market share to traditional field buses due to the already important existing base of ethernet cabling. We have used one of these protocols, EtherCAT, released by Beckhoff in 2003 [6], which is, according to manufacturers, able to exchange many distributed signals with cycle times below 100 μs [7].

Control Hardware

The hardware modules have been selected according to the demands imposed by the design specifications When the gap aperture is under 20 mm, the magnetic forces are very strong. Therefore special care has been taken in the selection of the motor drive to supply the needed torque when operating in microstepping operation at the target speed.

Summarising, the main hardware components of the PS and its control system include:

- A 5-phase Oriental Motor stepper motor (RK 564 AMCE) and its driver (RKD514LM-C).
- A Renishaw optical encoder (RGH22D) with 1 μm resolution and homing mark.
- Beckhoff PLC modules, (see Fig. 3) , listed in Table 1.
- 2 Limit switches, Saia T85 5E4

The first PS prototype used a 2-phase stepper motor and was controlled directly with a Beckhoff module without motor driver. Such philosophy was later abandoned due to excessive mechanical noise and motor vibration. After extensive testing, the EL7041 stepping motor controller

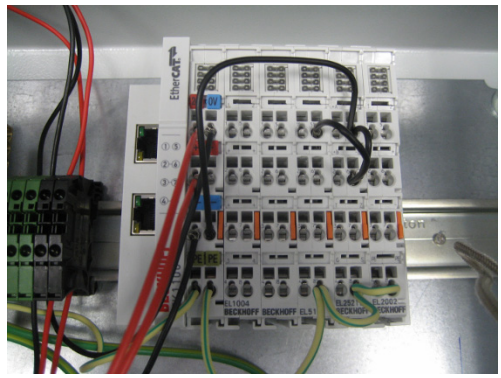


Figure 3: PS control modules before wiring.

Table 1: Beckhoff Modules Used

Module	Description
EK1100	EtherCAT coupler
EL1004	Digital inputs for limit switches
EL5101	Incremental encoder module
EL2521	Pulse train output for motor control
EL2002	Digital outputs for motor brake
EL3204	Terminal for temperature readings (PT100)

module was found to be responsible for this behaviour. A new release of this module, with an upgraded firmware and hardware, was tested afterwards with the same motor and the results were much better but still inferior to the option of equipping the PS with a motor and driver from the same manufacturer and using a Beckhoff pulse train digital output module to control the driver, a solution which gives a better performance and, subsequently, was adopted for the second PS prototype.

Software

The PS prototypes are controlled by a PC running Beckhoff Twincat PLC, that acts like a virtual machine running the PLC program. This program, developed at CIEMAT, has been coded in Structured Text following the IEC 61131-3 standard. Twincat PLC can run up to four PLC runtimes in a PC, and, specifically for motion control, Twincat PLC NC offers the TcMC2 library which is compliant with the PLCopen Motion Control standard [8]. CNC axes are referenced by the derived datatype AXIS_REF. Several function blocks (FB) are available for single or multiple axis control and reading their status variables.

As other projects were being developed at CIEMAT for EXFEL using stepper motors, it was decided to create a Twincat PLC library, MotorCommon.lib, to implement all these subsystems. This library uses a mix of basic function blocks available in the TcMC2 library and in-house developed functions better suited to the project needs. Any drive configured as an axis in Twincat PLC NC can be defined

as an instance of the *MotorControlWTimedBrakeAutoPow* function block. A wrapper (i.e. MoverMotor or PhaseShifterMotor) adapts the FB to any motor characteristics. For example, the PS stepper motor comes equipped with an electromagnetic brake and the motor windings can be activated or deactivated following an on/off signal. Due to the incremental nature of the position feedback, the PS wrapper defines also a homing procedure that has to be carried out when the device is started up or after an error. Every motor operational parameter, such as brake delay time, deadband position, backslash compensation, timeout and maximum operating temperature can also be configured.

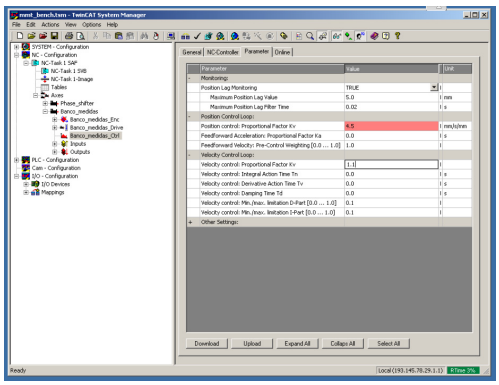


Figure 4: Axis configuration in Twincat System Manager.

The Twincat System Manager (see Fig. 4) is the configuration tool for the EtherCAT system, the PLC program, the I/O modules and the mechanical axes. A motor, or more generically a drive, can be defined as a Computer Numerical Control (CNC) axis and then linked to the corresponding modules. An axis has two basic elements, plus a third one: i) the actuator or drive, ii) the encoder or feedback, and iii) the controller or control policy. For instance, the PS axis is linked to both a pulse train output card (EL2521) and an incremental encoder card (EL5101). The controller can also be chosen and configured. The PS uses a position P control plus velocity PID with torque. The parameters for this controller have been manually tuned.

Although Twincat PLC comes with its own visualization tool, a classical SCADA GUI package, LabVIEW GUIs (see Fig. 5) have been developed in-house to take advantage of LabVIEW flexibility. The communication between LabVIEW and Twincat PLC has been achieved by using the Beckhoff TcADSdll.Dll library, which organises data exchange between Windows programs and Twincat. On top of that, a series of functions have been created to control the PLC from our GUI: start or stop the PLC, read variables and send data and commands. However this communication is asynchronous to the PLC cycle, forcing the Labview programm to constantly interrogate the TwinCAT server for changes in the status of signals. This could be a drawback for operation but does not pose any problem for testing and measurement. The validation of the series production will also be done automatically by a LabVIEW

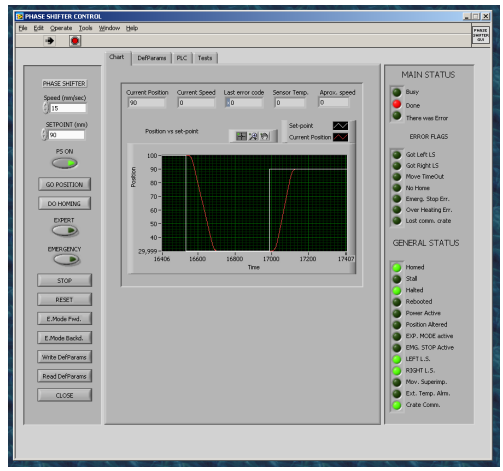


Figure 5: PS control LabVIEW GUI.

program, its development is under way.

RESULTS

The magnetic and mechanical measurements of the two prototypes have been reported in [9]. The first prototype achieved a repeatability of ($\pm 57\mu$m) in bidirectional movements, slightly over the limit, due to some mechanical problems that were corrected in the second prototype.

Some requirements have evolved over time, such as the gap opening speed. The latest request is that the PS shall follow the undulator, at speeds over 10 mm/s. Speeds up to 15 mm/s have been achieved with the second prototype. Preliminary measurements of this prototype, (see Fig. 6) show that positioning errors are within limits.

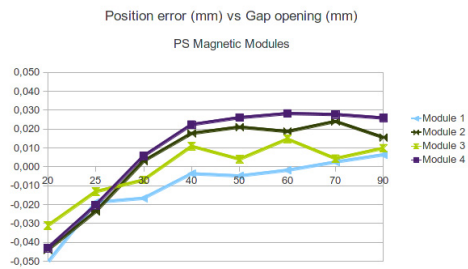


Figure 6: 2nd PS prototype preliminary measurements.

MAGNETIC MEASUREMENTS TEST BENCH

The Magnetic Measurements Test Bench (MMTB) (see Fig. 7) has been developed at CIEMAT as a tool to calibrate and measure the magnetic field of the Phase Shifter, in order to validate the magnetic design of the PS [10]. A linear actuator controlled by a dedicated Beckhoff module

moves a coil inside the PS gap. This coil is connected to a fluxmeter which integrates the magnetic flux, that is then read by a 24-bit ADC module to construct the $\overrightarrow{B}$ field integral after post-processing. The PLC controls both axes (PS and MMTB) and records position, speed and magnetic field values for further analysis. The position is monitored by an incremental rotational encoder. Its control software been written using the same libraries (PLC and LabVIEW) than the PS. Special care has been taken to synchronize the values that are written to a file by the PLC. Initially, after data analysis, a delay of a few milliseconds was found, dependent of the speed, so that in the Field Integral versus Position graph, the "in" cycle (when the coil is moved inside the PS) and the "out" cycle did not match. This is probably caused by the way the PLC works, as it reads the variables first and produces the timestamp at the moment of writing to the file.

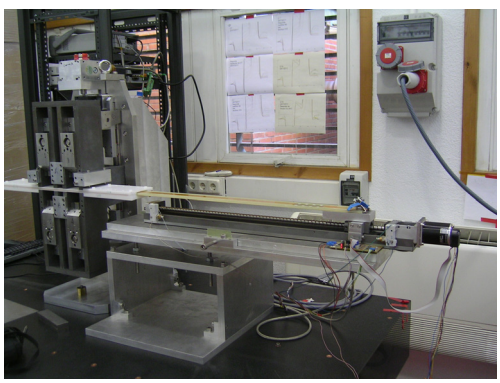


Figure 7: Phase Shifter and MMTB.

CONCLUSIONS

The Accelerator Technology Unit at CIEMAT is in charge of the design, manufacturing and delivery of several components to be installed in the 92 intersections of the Undulators System of the European XFEL, in particular, the Phase Shifter, the Quadrupole Mover and the Intersection Control Rack. All the control hardware has been implemented using commercial devices. In addition we have written a Beckhoff Twincat PLC library to encapsulate all devices as CNC axes within the Beckhoff architecture. GUIs have been created, based on an in-house developed LabVIEW library, able to communicate with the PLC runtime. As soon as the different prototypes have been available for testing in our labs, the integration of the control system has been quickly achieved. A fully automatic measurement procedure have been implemented. The first prototype achieved a repeatability of ($\pm 57 \mu$m), over the limit. The second prototype is now being tuned and the preliminary measurements have been satisfactory.

ACKNOWLEDGEMENTS

This work is supported by the CIEMAT and the Spanish Ministry of Science and Innovation under SEI Resolution of September 17th, 2009. The authors would like to express their gratitude to our colleagues at E-XFEL, mainly J. Pflüger and S. Karabekyan, for their encouragement and support during this project.

REFERENCES

[1] The European X-Ray Free Electron Laser (E-XFEL), http://www.xfel.eu.

[2] J. Pflüger, "Undulator Systems and Photon Diagnostics for the European XFEL Project", FEL 2005 Stanford, August 2005, TUOC002, p.378-382, http://jacow.org/f05/PAPERS/TUOC002.PDF.

[3] J.Munilla et al., "Design, Manufacturing and Tests of Closed-loop Quadrupole Mover Prototypes for European XFEL". Proceedings of the 2nd International Particle Accelerator Conference (IPAC), San Sebastian (Spain), September 2011

[4] H.H. Lu, et al., "The permanent magnet phase shifter for the European X-ray free electron laser", Nuclear Instrumentation and Methods, part A (2009), doi:10.1016j.nima.2009.03.217

[5] S.Sanz et al., "Evaluation of magnetic forces in permanent magnets". IEEE Transactions on Applied Superconductivity, vol 2010, doi:10.1109/TASC.2009.2038933

[6] D.Jansen, H.Buttner "Real-Time Ethernet: the EtherCAT solution". IET Computing Control Engineering, Feb 2004 vol.15 issue 1 http://www.iee.org/computingmagazine

[7] Technical Introduction and Overview, July 2005. EtherCAT Technology Group http://www.ethercat.org.

[8] PLCopen http://www.plcopen.org/.

[9] I. Moya et al., "Fabrication and testing of the first phase shifter prototypes built by CIEMAT for the European XFEL". Proceedings of the 2nd International Particle Accelerator Conference (IPAC), San Sebastian (Spain), September 2011

[10] S. Sanz et al., "Development of a Test Bench for the Magnetic Measurements on the Phase Shifters for the European XFEL", 22nd International Conference on Magnet Technology, Marseille (France), 2011

NSLS-II FILLING PATTERN MEASUREMENT*

Yong Hu#, Leo Bob Dalesio, Kiman Ha, Igor Pinayev, BNL, NSLS-II, NY 11973, U.S.A.

Abstract

Multi-bunch injection will be deployed at NSLS-II. High bandwidth diagnostic beam monitors with high-speed digitizers are used to measure bunch-by-bunch charge variation. The requirements of filling pattern measurement and layout of beam monitors are described. The evaluation results of commercial fast digitizer Agilent Acqiris and high bandwidth detector Bergoz FCT are presented.

INTRODUCTION

NSLS-II is designed to deliver photons with average spectral brightness in the 2 keV to 10 keV energy range exceeding 10^{21} ph/mm^2/mrad^2/s/0.1%BW [1]. This cutting-edge performance requires the storage ring to support a very high-current electron beam (500 mA) with sub-nm-rad horizontal emittance (~0.5 nm-rad) and diffraction-limited vertical emittance at ~8 pm-rad. To achieve this high-current (high intensity) requirement, NSLS-II will utilize a full-energy Injector which consists of a 200 MeV Linac, Linac to booster transport line (LtB), 3 GeV booster (BSR), booster to storage ring transport line (BtS). The Injector will operate in top-off mode and must supply ~ 7.3nC of charge once per minute. For single-bunch injection mode and a moderate repetition rate of a few Hz, replenishing this amount of charge would take a few seconds, occupying a significant fraction of the overall beam time. Therefore, multi-bunch injection (80~150 bunches) has been adopted, leading to minimal disturbance for user experiments.

In order to minimize intensity-correlated orbit oscillations due to uneven bunch patterns [2], we need to measure the filling pattern (also named bunch pattern or bunch structure) and then find a way to minimize bunch-to-bunch variation of current (or charge) if the variation exceeds 20%.

NSLS-II FILLING PATTERN

NSLS-II storage ring contains 1,320 RF buckets at 500 MHz. To alleviate the problems of ion trapping in the stored electron beam, approximately one-fifth of the buckets must be left empty. NSLS-II Injector will support uniform filling pattern in the storage ring. Two basic patterns were considered (two upper plots in Figure 1): uniform fill with the ion-clearing gap of about 20%; multiple uniform bunch trains with mini-gaps between them.

In addition to the nominal uniform fill, consideration is being given to specialized and complex bunch patterns: a single high-current bunch (named "camshaft" bunch, two lower plots in Figure 1) located in the middle of the ion-clearing gap; multiple camshaft bunches whose repetition rate is matched to the pulse format of modern pump lasers.

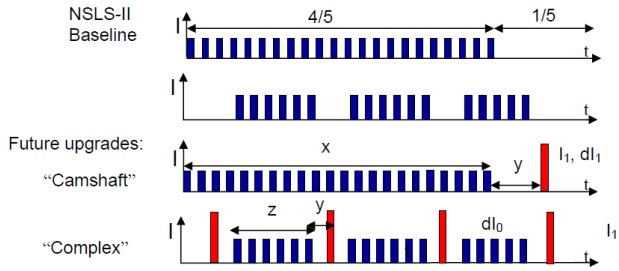


Figure 1: NSLS-II Filling Pattern [1]

One requirement [3] for NSLS-II filling pattern is that bunch-to-bunch charge variation should be less than 20%.

Filling pattern involves timing, diagnostics, injection / extraction, controls, etc. It will take efforts to implement arbitrary patterns for multi-bunch injection via filling pattern feedback. This paper focuses on filling pattern measurements: how to measure bunch structure and make this information available in EPICS [4]-based control system.

BEAM MONITORS

NSLS-II filling pattern measurement system provides relative bunch-to-bunch charge distribution along with the whole machine. This measurement requires combination of beam monitors (WCM, FCT, etc.), data acquisition (DAQ) and controls (fast digitizer, EPICS software, etc.) and Event Timing system [5]. Figure 2 shows the system architecture.

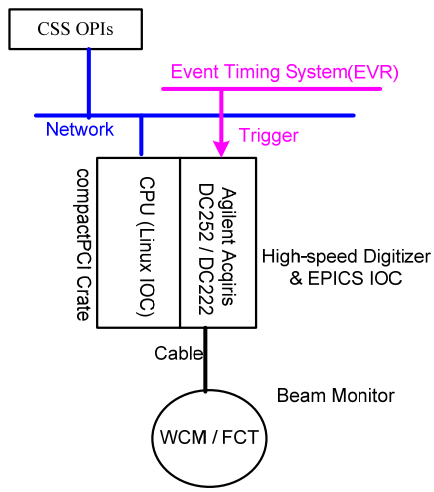


Figure 2: Overview of Filling Pattern Measurement

Since NSLS-II RF (Radio Frequency) is ~500 MHz, the bandwidth of beam monitors should be at least 500 MHz in order to distinguish individual bunch from the bunch-train (multi-bunch, 80~150 bunches). There will be 3 types of beam monitors distributed around the machine and all of them have very high bandwidth (>1 GHz): 5

* Work performed under auspices of the U.S. Department of Energy
# yhu@bnl.gov

wall current monitors (WCM) in Linac, 2 Bergoz FCTs (fast current transformer) in LtB, 1 FCT in BSR, 2 FCTs in BtS and 1 BPM (beam position monitor) button in Storage Ring.

Wall Current Monitor

The Linac diagnostics consist of five resistive WCMs to observe the longitudinal profile/bunch shape of the electron bunches after the gun, pre-buncher, and buncher. The WCM is formed by equally spaced broadband ceramic resistors mounted on a flexible circuit board, wrapped around a short ceramic break. The WCM has very high analog bandwidth at ~3GHz.

Fast Current Transformer

Commercial off-the-shelf Bergoz FCT, usually directly mounted on the beam chamber with a ceramic break and RF shielding, will provide electrical signal proportional to the charge of individual bunches. Its specification claims 1.75 GHz bandwidth with a 200 ps rise time. But our real measurements show only ~1 GHz BW which is still sufficient to get 500 MHz bunch-to-bunch information.

Beam Position Monitor

All BPMs in NSLS-II will utilize high precision 4-button pickup electrodes that are diagonally incorporated into the aluminium extrusion vacuum chamber. The diameter of BPM in the storage ring is only 7mm so that this kind of BPM has wide analog bandwidth (~ 2GHz) while minimum impact on the Ring impedance. The circulating current of 500 mA in the storage ring is expected to produce -1.1dBm signal into 50 Ω load at 500 MHz on BPM button. This signal is strong enough for filling pattern measurement.

CONTROLS

High-bandwidth filling pattern monitor requires high-speed digitizer to sample its analog output signal. According to the Nyquist sampling theorem, the minimum sampling rate of the digitizer required to discriminate 500 MHz bunch-to-bunch information is 1GS/s.

Hardware

Agilent U1065A Acqiris high-speed compactPCI digitizers (DC252 and DC222, 10-bit resolution, 2 GHz bandwidth, DC252 is 2-channel, 4 GS/s per channel, DC222 is 1-ch with 8GS/s) are selected for NSLS-II filling pattern measurement. Beam monitors associated with controls hardware in each sub-accelerator are listed in Table 1.

The cable length from the beam monitors to the digitizers is more than 100 feet. To reduce signal loss and attenuation, low loss RF cable LMR900 will be used. Although NSLS-II bunch length itself is very short at ~ 15 ps (RMS), the actual output signal from filling pattern monitor is Gaussian-shape pulse with ~ 1 ns width. In this case, 4 GS/s digitizer DC252 will provide at least 4 samples per bunch which should be sufficient to calculate individual bunch charge with appropriate algorithms such as Gaussian data-fitting and integrating samples' voltages.

Table 1: Filling Pattern Monitors & DAQ Hardware

Sub-accelerator	Beam Monitor	Digitizer
Linac	6 WCMs	3 DC252
LtB	2 FCTs	1 DC252
Booster	1 FCTs	1 DC222
BtS	2 FCTs	1 DC252
Ring	1 BPM	1 DC222

Software

NSLS-II beam diagnostics control system [6] will be completely based on EPICS. For OPI (operator interface) GUIs, CSS (Control System Studio) is our preference. The preferred operating systems are RTEMS (Real-Time Executive for Multiprocessor Systems) and Linux/Debian. For cPCI-based controls, the CPU board will be standardized as GE CT11 and diskless PXE booting Linux & EPICS driver is done by Debirf (DEBian on Initial Ram Filesystem [7]).

The EPICS driver for Acqiris digitizer is originally developed at SLAC. Several improvements are made at NSLS-II, including a few bug fixes and filling pattern related data processing.

Here's the basic implementation of our filling pattern measurement: the filling pattern EPICS IOCs will provide the following data: number of bunches (max. 150), normalized filling pattern (array data like filling[150]: 0.85, 1.00, 0.96, …), maximum bunch-to-bunch variation (i.e. $\Delta Q = 5\%$), individual bunch charge calibrated against ICT or DCCT or computed from integral voltage signal. If measured $\Delta Q < 20\%$, our job is done; If $\Delta Q >= 20\%$, we need to find out what's the cause and find a way to minimize ΔQ.

Here's our procedure to compute normalized filling pattern inside EPICS IOC software:

1) Search for positive (or negative)pulse peaks with peak threshold;
2) Gaussian-fitting on 5 samples (2 samples before the peak, 2 after the peak) and then integrate(sum) them;
3) Find the max. integrating value;
4) Normalize all integrating values against the max.

BENCH TESTS

We have performed bench tests on Bergoz FCT and Acqiris digitizer.

Bench Tests on FCT

We have purchased 5 Bergoz FCTs and performed acceptance tests. We have measured FCT bandwidth using network analyzer. Figure 3 shows that one of the FCTs has ~ 1GHz BW (-3dB).

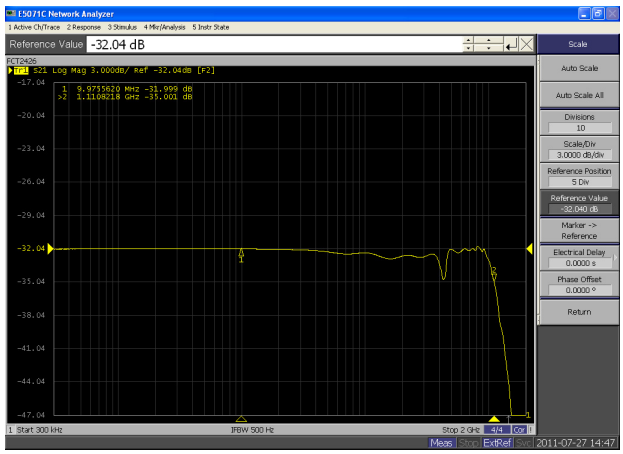


Figure 3: FCT Bandwidth Measurement

Bench Tests on Acqiris

We have completed performance evaluation on the ultra fast digitizer Acqiris DC252, including max. sampling rate at 8 GS/s by combining 2 channels to 1 channel (interleaving), effective number of bit (ENOB), etc. The input test signal is 500MHz sine wave, 0.9Vpp.

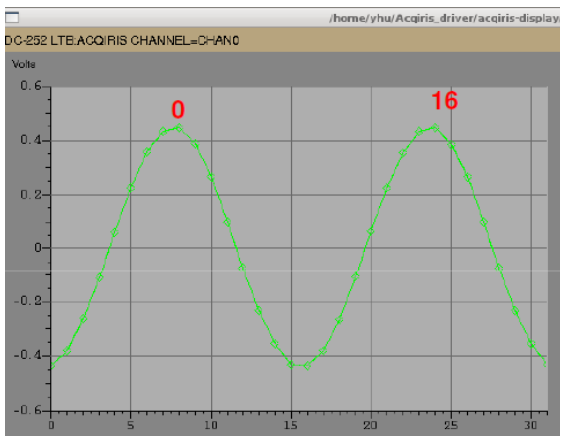


Figure 4: 8GS/s acqiris Digitizer.

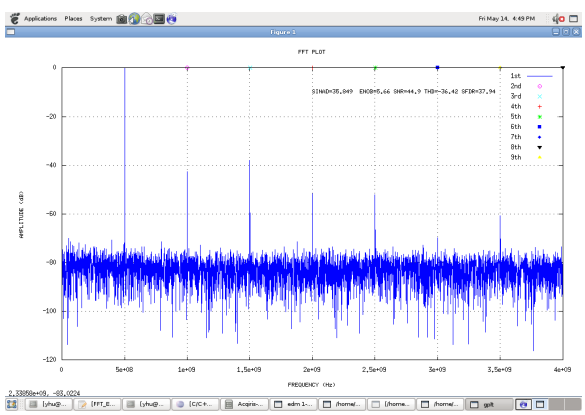


Figure 5: 7-bit ENOB after Sine-wave Data-fitting.

FILLING PATTERN SIMULATION

In order to test our algorithm software for filling pattern measurement as well as interface to Event timing system, we have setup a test stand to simulate filling pattern as shown in Figure 5.

- Timing EVG/EVR: provides 2 triggers, one is for triggering digitizer, the other for triggering pulse generator;
- pulse generator DG645: receive external trigger from EVR and then use burst mode to generate multiple pulses with 2ns width and 10MHz rate;
- FCT: input signal from DG645 and output to digitizer;
- Acqiris digitizer: triggered by EVR and acquire FCT signal;

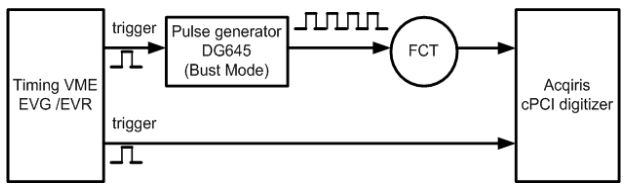


Figure 6: Filling Pattern Simulation.

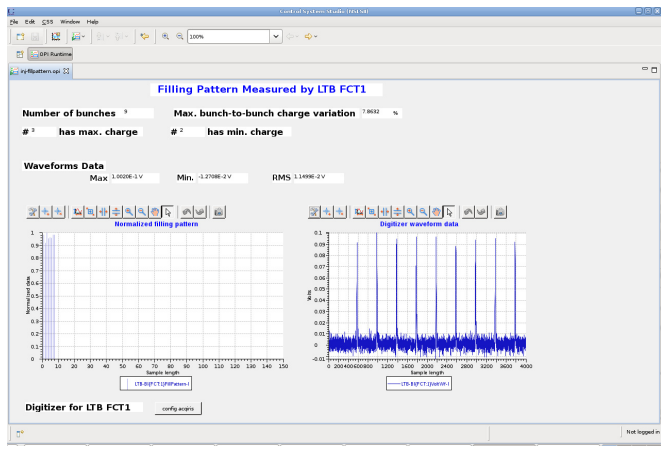


Figure 7: Filling Pattern Simulation Result.

CONCLUSIONS

Filling pattern measurement system is well understood and the controls hardware & software are well tested. We're ready for NSLS-II Injector installation and commissioning.

REFERENCES

[1] NSLS-II Preliminary Design Report", http://www.bnl.gov/nsls2/project/PDR.
[2] [sls2] http://slsbd.psi.ch/pub/slsnotes/sls1697.
[3] Yong Hu, et al., "Diagnostics Control Requirements and applications at NSLS-II", Proceedings of ICALEPCS 2011, Grenoble, France.
[4] http://www.aps.anl.gov/epics/.
[5] http://www.mrf.fi/.
[6] Yong Hu, et al., "NSLS-II Beam Diagnostics Control System", Proceedings of ICALEPCS 2011, Grenoble, France.
[7] http://cmrg.fifthhorseman.net/wiki/debirf.

A NEW HELMHOLTZ COIL PERMANENT MAGNET MEASUREMENT SYSTEM*

Joseph Z. Xu and Isaac Vasserman, ANL, Argonne, IL 60439, U.S.A.

Abstract

A Helmholtz coil magnet measurement system was upgraded at the Advanced Phone Source (APS) to characterize the insertion device permanent magnets [1]. The system uses the latest state-of-the-art field programmable gate array (FPGA) technology to compensate the speed variations of the magnet motion. Initial results demonstrate that the system achieves a measurement precision better than 2×10^{-4}.

INTRODUCTION

Most insertion devices (IDs) at the APS are permanent magnet based [2]. Before the magnets are installed onto the IDs, each of them has to be characterized and sorted to ensure that the magnets in a single ID are even and as identical as they can be to minimize the integral fields and the phase errors of the device.

The APS Helmholtz coil system, designed to characterize magnets used in the APS permanent magnet IDs, has been recently upgraded. The system has improved the measurement precision by an order of magnitude. The system now consists of a pair of identical coils, a horizontal rotary table with two full ceramic spherical bearings, and a digital encoder on its main rotating shaft driven by a servo motor.

The control and data acquisition system is a PXI-based computer system. With the latest state-of-the-art field programmable gate array (FPGA) technology, the system is capable of synchronized measurements of position (0.05 degree in resolution), time (25 ns), and voltage (16 bit).

THEORY OF OPERATION

A Helmholtz coil consists of two identical circular coils placed coaxially and separated by a distance equal to the radius of the coils. When the two coils carry the same current in the same direction, it creates a near uniform magnetic field within the center region between the two coils. Therefore, a magnet placed within the center region can be treated as a magnetic dipole by a Helmholtz coil system.

Moving a magnet inside the coils causes the magnetic flux change that in turn generates a voltage signal in the coil:

$$V = -d\Phi/dt\,, \tag{1}$$

where V is the voltage, Φ is the magnetic flux, and t is time.

By rotating the magnet inside the Helmholtz coils, measuring the voltage signal in the coils, and integrating the signal over time, we have [1]

$$\Phi = -\int_{t_0}^{t_1} V(\theta.t)dt$$

$$= -\frac{1}{k} * (M_V * \sin\theta + M_H * \cos\theta) + C, \tag{2}$$

where V is the voltage signal, θ is the rotation angle, k is the Helmholtz coil constant, M_V and M_H are the vertical and horizontal components of the magnetic moment, respectively, and C is the offset constant.

$$k = \left(\frac{5}{4}\right)^{\frac{3}{2}} \frac{R}{\mu_0 N}\,, \tag{3}$$

where R is the coil radius, μ_0 is the free space permeability constant, and N is the number of turns in each coil.

Fitting the integral with a sinusoidal function of

$$A * \sin\theta + B * \cos\theta + C \tag{4}$$

we have the vertical component of the magnetic moment:

$$M_V = kA \tag{5}$$

and the horizontal component of the magnetic moment:

$$M_H = kB. \tag{6}$$

Usually magnet vendors provide the components of the intrinsic induction information.

$$B = \mu_0 m = \frac{\mu_0 M}{V}, \tag{7}$$

where m is the magnetization vector and V is the volume of the magnet. Hence the components of the intrinsic induction are

$$B_{V,H} = \mu_0 m_{V,H} = \frac{\mu_0 M_{V,H}}{V}. \tag{8}$$

SYSTEM DESCRIPTION

The Helmholtz coil magnet measurement system consists of the subsystems shown in Figure 1.

A set of Helmholtz coils that consists of a pair of identical coils are made of copper material and the support structure is of G-10. The mean diameter of the coils is 26 inches. The coils are mounted coaxially, separated by a distance of 13 inches. Each coil has 392 turns.

A horizontal rotary stage holds the magnet in the center region of the coils. The aluminum stage shaft is supported by two full ceramic spherical bearings. A magnet mounting stage is supported at the end of the aluminum shaft. The mounting stage is centered between the two coils. The stage is motorized, driven by a servo motor that coupled to the other end of the stage via a 10:1 rotary inline gearbox. A hollow-shaft rotary encoder is

* Work at Argonne was supported by the U. S. Department of Energy, Office of Science, Office of Basic Energy Sciences under Contract No DE-AC02-06CH11357.

Figure 1: Helmholtz coil magnet measurement system.

mounted inline on the stage shaft. The encoder has 360 degrees freedom with 0.05-degree resolution. The rotary encoder is used to accurately define the angular position of the magnet mounted on the rotary stage.

The FPGA reconfigurable data acquisition card has eight 16-bit-resolution analog inputs, 96 digital inputs/outputs, 25-ns time resolution, and 80-kB on-board memory. The digital I/Os are programmed to read the encoder positions. The analog inputs are configured to measure the coil signals synchronized with the position readouts and the time durations in real time.

A PXI shelf with a control card hosts the FPGA card and the software program.

SYSTEM CONTROL SOFTWARE

LabVIEW-based system software has been developed to coordinate the stage control and data acquisition. Figure 2 shows the schematic layout of the system control and data acquisition architecture. The system can be accessed via the Internet from anywhere anytime, wired or wireless, through the embedded http interfaces inside LabVIEW.

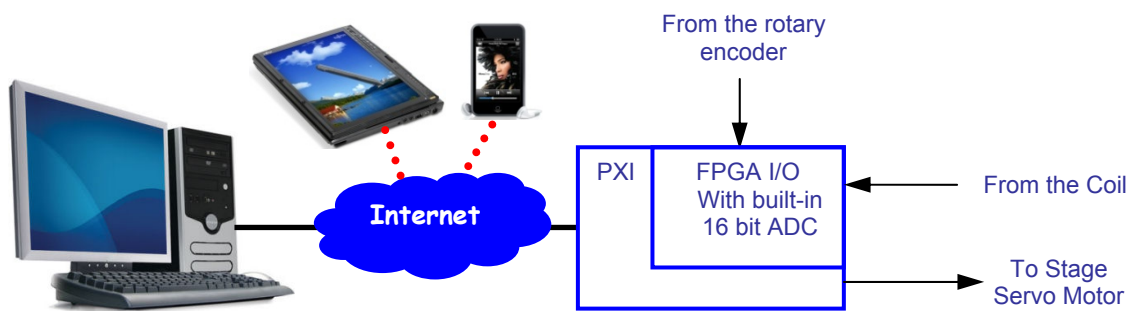


Figure 2: System control and data acquisition architecture schematic layout.

The system software has a main control and data acquisition interface and modules. The main control and data acquisition interface rotates the magnet stage and sets the FPGA hardware to wait for the magnet angular position to go to the start position. When the angular position hits the start position mark, the hardware will simultaneously write the following data to its memory on the fly:

1. The angular positions of the magnet with an angular resolution that can be set down to 0.1 degree per step.
2. The integrated Helmholtz coil voltage signals across each angular step with a rate of 4.3 μs per sample at 16-bit resolution.
3. The number of samples integrated within each angular step.
4. The time duration within each angular step.

When the magnet angular position reaches the 360 degree mark, the hardware sends an interrupt to the module. It integrates the signal, fits the integrated data with a sinusoidal function to extract the field integral components, and plots the raw data along with the fittings.

The measurement analysis and plot module reads the saved data file, analyzes the data, and plots the raw data along with the fittings and analysis results.

The advanced motion control module fine tunes the stage position and encoder readout. It has one submodule: the motor velocity and acceleration control and monitoring. The submodule sets and monitors the velocity and acceleration parameters of the servomotor.

The system parameter database module consists of three copies in a single file: the current, the backup and the default. The default copy is read-only. Each copy contains, among others, the motor velocity, acceleration and system calibration constants, encoder reference indexes, encoder resolutions, stage-gear ratio, motor-velocity ratio, motor-acceleration ratio, and motor resolutions. The system parameter database control module and its advanced system parameter database control module manipulate and manage the system parameter database.

The integral measurement FPGA firmware module resides on the FPGA reconfigurable card. It monitors the encoder position, the coil voltage, and time. It also interfaces with the measurement modules and submodules. The FPGA module executes all the tasks in parallel at the speed of 40 MHz.

The main control and data acquisition interface, each module, and submodule has its own GUI interface except for the FPGA firmware module. The main interface provides access to all the modules and the submodules. It checks the status of the FPGA reconfiguration data acquisition. If the FPGA card is not initialized or is running on different firmware, the module will download and initialize the card with the appropriate firmware. It also checks the status of the servomotor. If the motor is not initialized, it will try to reinitialize it.

SYSTEM CALIBRATION

The system calibration is the measurement of the Helmholtz coil constant k as defined in equation (3). Based on the design of R = 13 inches and N = 392, the k value shall be 0.09362 Ampere per Gauss. However, the real system k value has to be calibrated.

Hardware required for the calibration is a calibrated current power supply and a calibrated magnetometer. The DC current power supply used for calibration was a Kaithley 6221 DC current source. The magnetometer was a Bartington Instruments Mag-03 Magnetic Field Sensor. The calibration results are shown in table 1.

Table 1: Calibration Results

Field (G)	+100 mA	-100 mA
Mean value	1.081190	-1.081188
Standard deviation	0.000058	0.000032

Therefore, the k constant is 0.09249 Ampere per Gauss, which is very close to the design value of 0.09362 Ampere per Gauss.

MEASUREMENT RESULTS AND DISCUSSION

A typical magnet measurement taken with the Helmholtz coil measurement system is shown in Figure 3.

The file header field shows the basic parameters of the measurement. The rotary stage scanning speed was 2 revolutions per second. Each measurement consists of scans along the +y axis (+Mx/+Mz), +x axis (+My/+Mz), -y axis (-Mx/+Mz), and –x axis (-My/+Mz). Averaging over the components cancels the mechanical asymmetries of the system. The integral plot field shows a specific integrated raw scan along with its sinusoidal fitting. Next to the integrals field are the fitting parameters to that specific measurement. According to equations (5) and (6), the sine component represents the vertical component of the field integral while the cosine component represents the horizontal component. Below the integral field plot display are the magnetic moment components $m_{(ave)}$, the measurement errors $\sigma_{(err)}$, the moment density components $\rho_{(density)}$, the magnetic moment orientations $\theta_{(angle)}$, as well as the total moment M, and its orientation to the z axis θ_z.

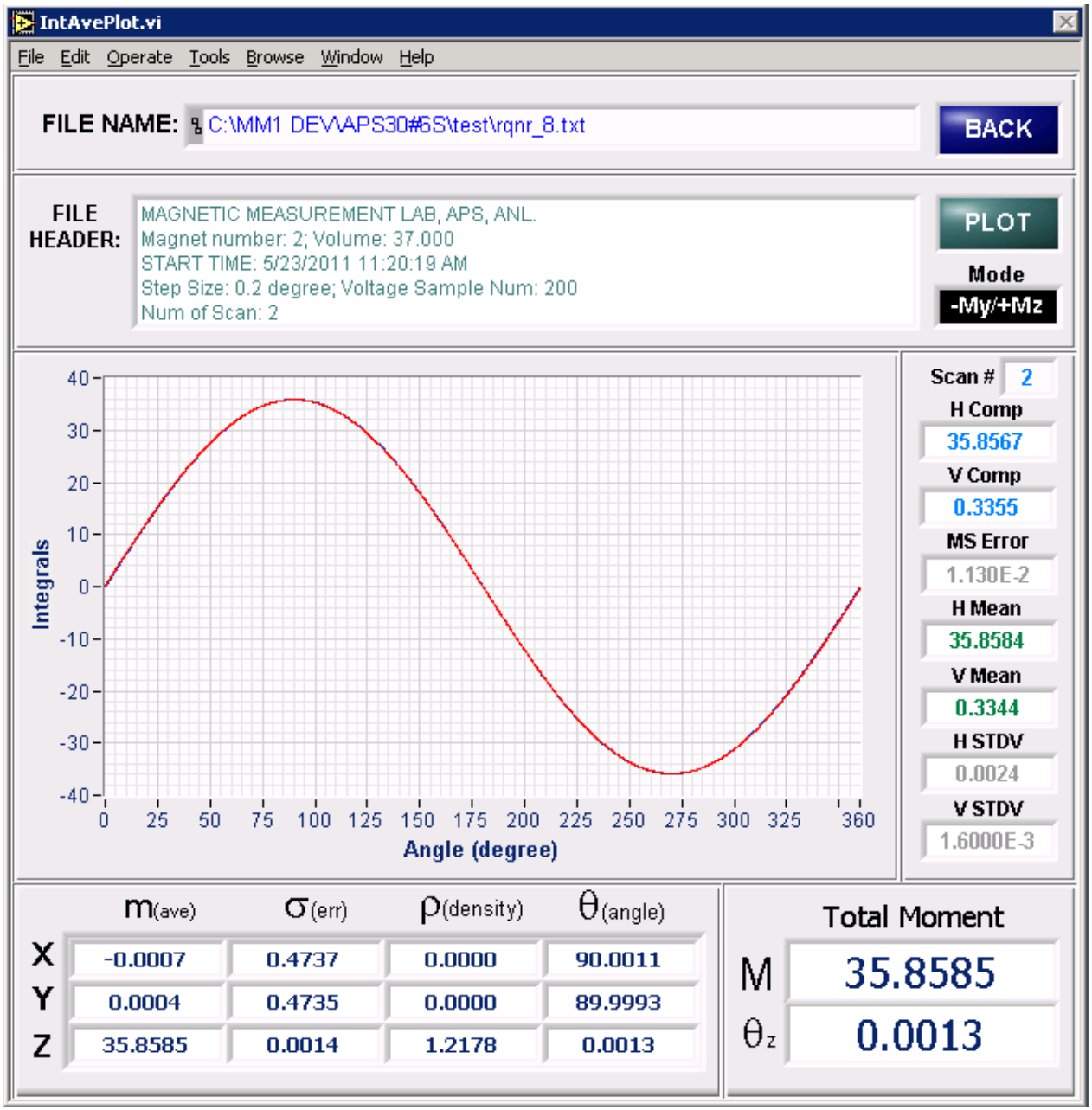


Figure 3: Magnet measurements.

CONCLUSION

A new Helmholtz coil magnet measurement system has been constructed, tested, and commissioned at the Advanced Photon Source. With the latest state-of-the-art FPGA technology, the system achieves measurement precision of 2×10^{-4}.

REFERENCES

[1] D.W. Carnegie and J. Timpf, "Characterizing permanent magnet blocks with Helmholtz coils," Nucl. Instrum. Methods A 319 (1992) 97-99.

[2] L. Burkel, R. Dejus, J. Maines, J. O'Brien, J. Pflueger, I. Vasserman, "The Insertion Device Magnetic Measurement Facility: Prototype and Operational Procedures," ANL/APS/TB-12, March 1993.

NSLS-II VACUUM CONTROL FOR CHAMBER ACCEPTANCE

H. Xu, L.R. Dalesio, M. Ferreira, H. Hseuh, D. T. Zigrosser, Photon Science Directorate, BNL, Upton, NY 11973, U.S.A.

Abstract

The National Synchrotron Light Source II (NSLS-II) uses extruded aluminium chambers as an integral part of the vacuum system. Prior to installation in the Storage Ring all dipole and multipole chamber assemblies must be tested to ensure vacuum integrity. A significant part of the chamber test requires a full bakeout of the assembly, as well as control and monitoring of the titanium sublimation pumps (TSP), non-evaporable getter pumps (NEG) and ion pumps. Data that will be acquired by the system during bakeouts includes system temperature, vacuum pressure, residual gas analyzer (RGA) scans, ion pump current, TSP operation and NEG activation. This data will be used as part of the acceptance process of the chambers prior to the installation in the storage ring tunnel. This paper presents the design and implementation of the vacuum bakeout control, as well as related vacuum control issues.

INTRODUCTION

The NSLS-II storage ring is made up of 30 cells and 30 straight sections. Each vacuum cell consists of five basic chambers; an upstream matching multipole chamber, a dipole chamber, a high dispersion section multipole chamber, a second dipole chamber, and a downstream matching multipole chamber. There are RF-shielded bellows connecting these chambers together. The straight sections, either 6.6 m or 9.3 m long between the cells are for insertion devices and for special sub-systems such as RF cavities, injection devices, , and so forth. [1]

Most of the NSLS-II storage vacuum chambers are made of extruded aluminium with specific cross-sections. These vacuum chambers allow in-situ bakeouts to 150˚C to help remove absorbents, such as water, and contaminants on the inner surface.

VACUUM CHAMBER BAKEOUTS

The vacuum chambers undergo at least two bakeout cycles prior to their installation in the storage ring. Once the chambers are instrumented with vacuum components such as NEG strips, small ion pump, gauging and RGA, they are prepped for the first vacuum bakeout. A turbomolecular pump backed by a dry mechanical pump is used to pump down the vacuum chamber and check for leaks using a helium leak detector. The vacuum level and residual gas composition are monitored during the bakeout using vacuum gauges and residual gas analyzer. The entire chamber will be baked at 150˚C for ~ 16 hours to remove absorbed water and other contaminants from the surfaces. At this stage most vacuum controllers are manually operated. RGA readings are logged with a local Windows computer. The NEG strip is powered and controlled by a DC power source with degassing and activation of NEG done manually.

The second vacuum bakeout occurs at the end of girder integration, after the installation and alignment of the magnets and chambers on the girder assembly, and with the integration of peripheral components, such as large ion pumps, TSP, photon absorbers, and vacuum gauges. A typical girder assembly ready for bakeout is shown in Figure 1.

Figure 1: Typical NSLS-II girder assembly

The final bakeout will be carried out in storage ring tunnel after the installation and alignment of individual chamber/magnet girders in the tunnel, the connection of the five girder chambers with RF shielded bellows and the mounting of the RF shielded gate valves at the ends of the cell. It's planned to take a similar approach to the bakeout of an entire cell in the storage ring. This paper focuses on girder bakeouts.

VACUUM GIRDER ASSEMBLY BAKEOUT

There are several variations of girder assemblies; hence the bakeout system must be flexible and adaptable to accommodate different zone configurations.

Each girder assembly is divided into several bakeout zones. The heating of the chamber itself is accomplished by the use of a rod heater inserted into a dedicated channel along the chamber length. Appendage components are heated using custom silicon heating jackets. All heating zones are controlled by a multiple zone bakeout system, a BNL design incorporating 12 or 24 Omega signal zone temperature controllers with solid state relays housed in a common chassis. Over temperature interlocks are provided to protect chamber assemblies from overheating. All temperature controllers are connected through RS-485 links. The 24 zone carts

will also be used for storage ring cell bakeout where up to 3 carts are needed for each cell.

The vacuum chamber is pumped down with a turbomolecular pump backed by a dry mechanical pump during the bakeout. The vacuum level and gas composition are monitored continuously using vacuum gauges and RGA during the bakeout. The temperature of the chamber is raised slowly by the cal-rod heater, and controlled by the programmable temperature controller with inputs from the installed type E thermocouples on the chamber surface. A ~ 2.5 hour ramp period is programmed to reach the designed temperature value. The entire vacuum chamber is then held at 150˚C for ~12 hours to remove absorbed water and other contaminants from the surfaces. The sputter ion pumps, titanium sublimation pumps, and NEG strips are degassed during the soak period.

Table 1 shows all the components and controllers used for one girder baking. The number may vary with different girder. The vacuum device controllers used for one girder are assembled in one rack.

Table 1: Typical Vacuum Components and Controllers of Girder Bakeout System

Cold Cathode Gauge (CCG)	2
Convection Pirani Gauge (TCG)	1
Vacuum Gauge Controller (VGC)	1
Sputter Ion Pump (IP)	2
Ion Pump Controller (IPC)	1
Titanium Sublimation Pump (TSP)	1
Titanium Sublimation Pump Controller (TSPC)	1
Non-Evaporable Getter Strip (NEG)	1
DC Power Supply for NEG Strip	1
Bakeout Cart (BC)	1
Residual Gas Analyzer (RGA)	1
Turbomolecular Pump Station (TMP)	1
I/O Controller	1
Data Server	1
Uninterruptible Power Supply(UPS)	1

CONTROL FOR CHAMBER BAKING

Overview

The core of the Experimental Physics and Industrial Control System (EPICS) [2] has been chosen as the basis for NSLS-II control system. Figure 2 shows the architecture of the EPICS based vacuum bakeout control system.

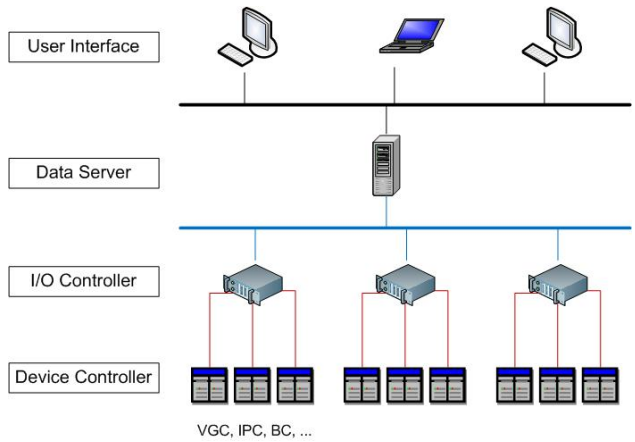


Figure 2: Control architecture for girder assembly bakeout.

Input and Output Controller (IOC)

Considering that the parameters to be monitored and controlled are much less than 1000 for girder bakeout system, we adopted soft IOC which runs on a Moxa DA-710 [3]. The Moxa works as both the IOC and terminal server. The DA-710 is based on the Intel x86 processor and runs on pre-installed Linux platform. It comes with 4 PCI slots for expansion modules and supports up to 32 serial ports.

Operator Interface

All of the monitor and control panels are created with software tools in Control System Studio (CSS) [4], which is an Eclipse-based collection of tools to monitor and operate large scale control systems.

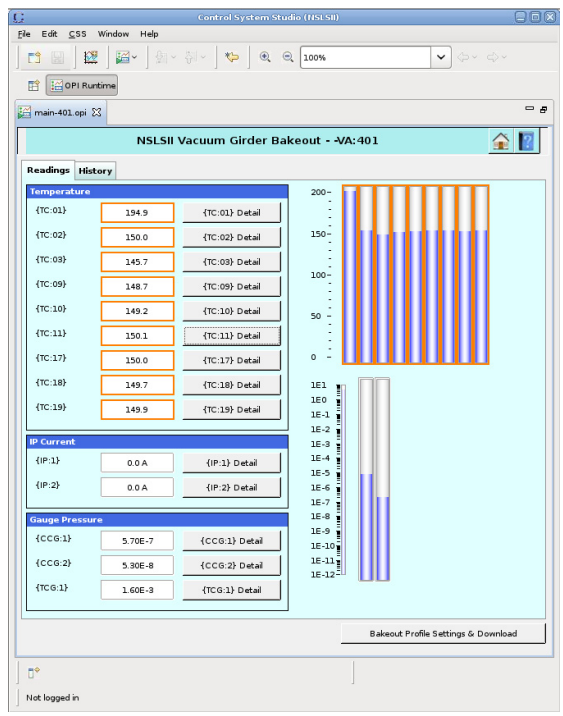


Figure 3: Main display screen for temperature readings and pressure readings.

A user-friendly interface was developed based on working experience with vacuum operators and engineers. The interface allows the operators to edit the desired ramp/soak profiles and provides the option to use predefined recipes. All zone temperatures, and gauge pressure readings are shown on main display panel as shown in Figure 3. Currently only local controls to the vacuum components are required. For future tunnel bakeout, remote controls will be required and provided.

Data Logging

Channel Archiver is used to log all temperatures, ion pump current, and gauge pressure readings. A low-cost Acer Aspire Revo series PC is used as the data server. This PC also works as the gateway between campus computers and I/O controllers due to network security issues. Users on the campus network are required to log onto the gateway computer to gain access to the girder bakeout system.

Data browser provides an easy way to review and analyze logged data. Figure 4 shows the historical view of the temperature readings.

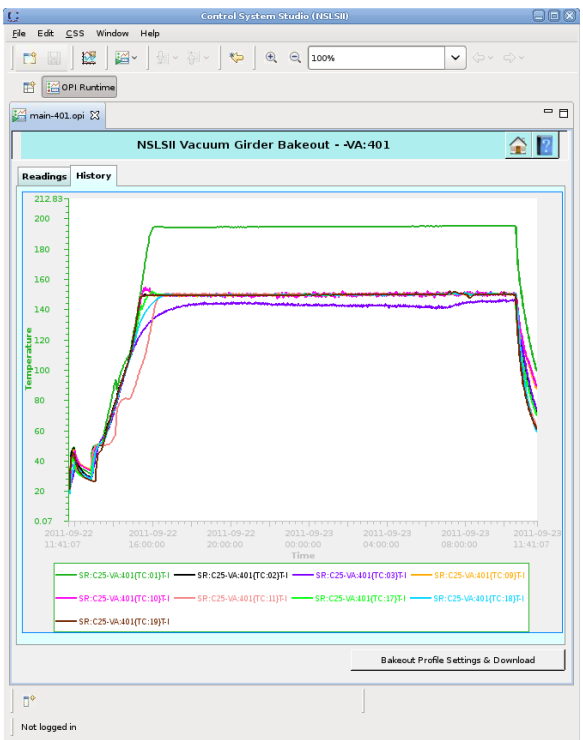


Figure 4: Historical data with data browser.

Interlock

It is necessary to have vacuum interlocks to protect vacuum assemblies from over pressure. The interlock is provided by a MKS 937B gauge controller. A Cold Cathode and Pirani gauge pair provides an adjustable setpoint that is hardwired to each of the vacuum controllers including the ion pump controller, tsp, bakeout cart and NEG controllers. The setpoint status can be monitored via the serial communication port of the gauge controller.

SUMMARY

To date, 4 fully assembled girder chambers have been processed and successfully baked. The girder bakeout control system has worked well. We're still learning from the girder bakeout and intend on improving system performance.

REFERENCES

[1] NSLS-II Preliminary design report, 2008, http://www.bnl.gov/nsls2/project/PDR.
[2] EPICS, http://www.aps.anl.gov/epics.
[3] Moxa, http://www.moxa.com.
[4] CSS, http://cs-studio.sourceforge.net.

A PROGRAMMABLE LOGIC CONTROLLER-BASED SYSTEM FOR THE RECIRCULATION OF LIQUID C_6F_{14} IN THE ALICE HIGH MOMENTUM PARTICLE IDENTIFICATION DETECTOR AT THE LARGE HADRON COLLIDER

I. Sgura *, G. De Cataldo, A. Franco, C. Pastore, G.Volpe,
INFN sez. Bari and Dep. of Physics Via G. Amendola 173, 70126 Bari Italy

Abstract

The aim of this paper is to present the design and the implementation of the Control System (CS) for the recirculation of liquid Perfluorohexane (C_6F_{14}) for the ALICE High Momentum Particle IDentification detector (HMPID).

The HMPID is a detector of the ALICE experiment at the CERN Large Hadron Collider (LHC). It uses liquid C_6F_{14} as Cherenkov *radiator* medium in twenty-one quartz vessels for the measurement of the charged particles velocity.

The primary task of the Liquid Circulation System (LCS) is to ensure the highest transparency of C_6F_{14} to the ultraviolet light. In order to provide safe long term operation a Programmable Logic Controller-based CS has been implemented.

CS provides both automatic and manual operating modes, remotely or locally. Its finite state machine design minimizes the possible operator errors and provides a hierarchical control structure allowing the operation and monitoring down to a single *radiator* vessel. LCS is protected against unsafe working conditions by both active and passive measures. The passive ones are intrinsically guaranteed whereas the active ones are ensured via the control software running in the PLC. The human interface and data archiving are provided via PVSS, the Supervisory Control And Data Acquisition (SCADA) framework which integrates the full detector control.

LCS under CS control proved to meet all designed requirements thus enabling HMPID detector to successfully collect data since the very beginning of LHC operation.

INTRODUCTION

The ALICE-High Momentum Particle IDentification detector (HMPID) is a proximity focusing Ring Imaging CHerenkov (RICH), for the identification of charged pions and kaons in the range of $1 < pt < 3$ GeV/c and protons in the range $2 < pt < 5$GeV/c [1]. It consists of seven identical RICH modules exploiting large area CsI photocathodes for Cherenkov light imaging [2]. HMPID uses liquid C_6F_{14} as Cherenkov *radiator* medium in the twenty-one quartz vessels coupled to Multi-Wire Chambers (MWPC) equipped with pad segmented CsI photo cathodes.

LCS [3,4] is a closed, pressure-regulated apparatus which purifies, fills, re-circulates and empties the *radiator* vessels (Fig. 1). The Intrinsically safe working conditions are ensured by a gravity flow circuit preventing any accidental over-pressure.

Due to LCS complexity and its inaccessible location during the LHC operation, its safe long term operation is ensured then by a dedicated Control System (CS). This latter is integrated in the HMPID Detector Control System (DCS) [5, 6].

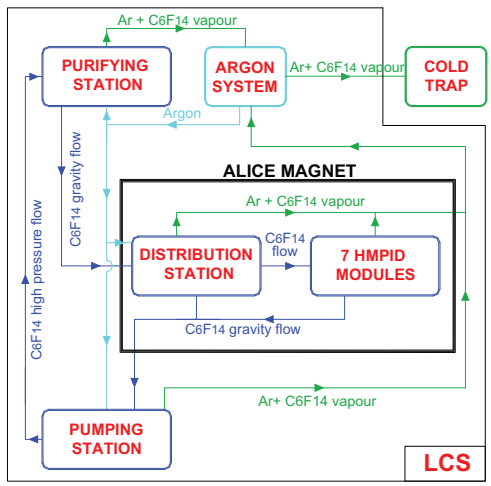


Figure 1: Block diagram of LCS: it consists of distinct stations interconnected and located in ALICE experimental area. The liquid is continuously pumped from the pumping station to the purifying station from where the liquid flows by gravity into the distribution station. This latter controls the flow rate in the *radiator* vessels from which the liquid returns in the pumping station where the cycle restarts.

THE C_6F_{14} CONTROL SYSTEM

Since 2003 C_6F_{14} CS has been conceived and implemented to fulfil the following requirements:

- to be highly reliable and scalable, as well as simple and robust;
- to be coherent and homogeneous with all detector control sub-systems;
- to be operable permanently and independently of the state of the other HMPID subsystems;
- to be able to take immediate actions for protecting all sensitive system components.

*corresponding author: irene.sgura@ba.infn.it

To fulfil these requirements a control system based on Siemens S7-300 Programmable Logic Controller (PLC) has been then chosen.

Architecture

According to the industrial automation standards ANSI/ISA 95, in C_6F_{14} CS architecture can be recognised three hierarchical levels: field level, control level and supervisory level (Figure 2).

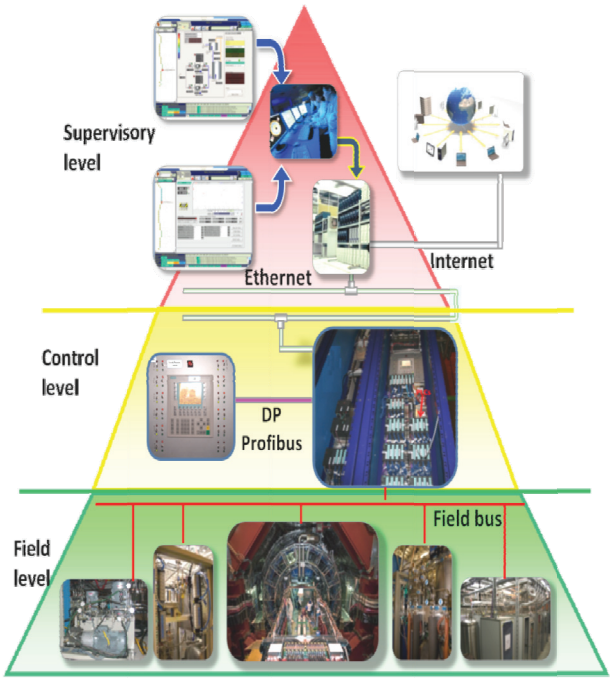


Figure 2: C_6F_{14} CS hardware architecture.

LCS represents the first level with all used sensors and actuators that work with industrial standard current and voltage ranges.

Control Level

The control level is the core where the control software processes run. It controls the underlying sensors/actuators by receiving and sending information through field buses.

Its design has been oriented towards the definition of elementary units (RICH $_{ith}$ 1< i <7, purifying and pumping stations) which can be controlled and monitored singularly in LCS. Figure 3 shows a schematic of the Rich unit which corresponds to one HMPID module.

The hardware of the control level consists of a network of Siemens S7-300 PLCs, one per control unit, reflecting the traditional Master-Slave requirements. They are interconnected using the PROFIBUS DP [7] communication protocol.

The Master PLC acts as gateway between the supervisory and control level (Figure 4).

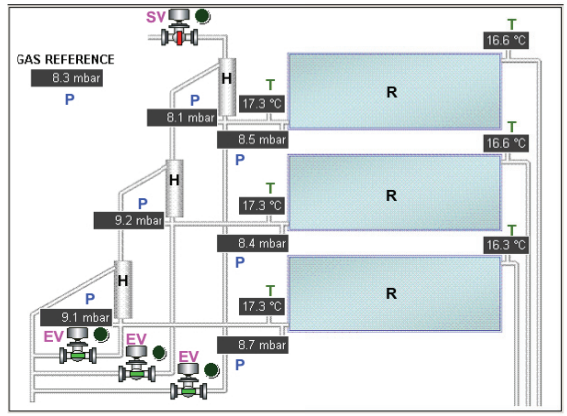


Figure 3: RICH$_i$ elementary unit: it consists of the i.th HMPID module's three *Radiator* vessels (R), the Header tubes (H) for supplying the matching *radiators*, the Pressure transducers (P), the Temperature sensors (T), the Supply and Emptying Valves (SV/EV) needed for module's operations.

This hardware architecture ensures:

- insulation of the control structure by the gateway to the external environment;
- remote implementation of the system's semantic verification procedure. Even if the operator commands are correct, the gateway verifies their compatibility with LCS state;
- local access via PROFIBUS DP by the Panel Operator LP (Figure 4), located in an accessible area during LHC operations. LCS can continue to work even if the supervisory level does not work. Both the local and remote control can be blocked by an hardware switch on LP, still ensuring monitoring.

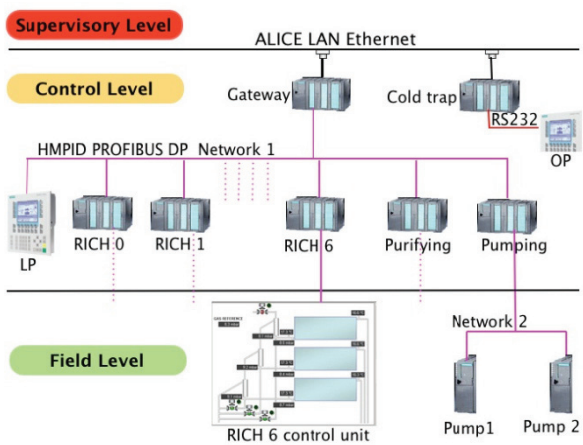


Figure 4: Hardware architecture of the control level.

The control system was developed in the environment Siemens STEP 7 [8] through the software tool Simatic Manager.

FSM Modelling

In order to automate LCS operation, the behaviour of each elementary unit has been modelled as a Finite State Machine (FSM), using Simatic S7 Graph language [9].

The unit behaves according to a state diagram and can move between defined states by means of executing actions. These actions are triggered either by the operator or by other events such as state changes of defined control-process.

According to the Cern JCOP prescriptions [10] three increasing severity alarm levels have been identified: WARNING, ERROR and FATAL.

Both the WARNING and ERROR allow the system to continue the normal operation. The FATAL level alarm is active when harmful conditions happen. In this case the CS has been designed to react automatically preventing permanent damages. In order to provide some flexibility during the commissioning phase, the automatic and manual operational modes are defined. In MANUAL state, the operator can directly check all the actuators, under its own responsibility, while the safety checks are always active.

In Figure 5 the state diagram of the RICH$_i$ FSM is shown.

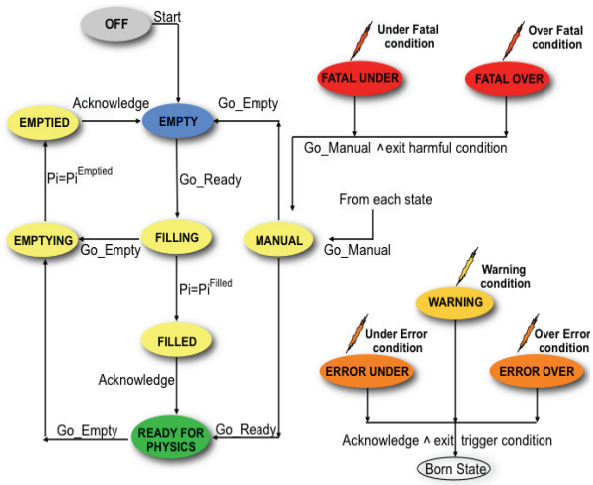


Figure 5: State diagram of the RICH$_i$ FSM. In "Ready for Physics" the *radiator* vessels are filled, the recirculation is active. A "FATAL OVER" condition rises when *radiator* vessel's pressure reaches 140 mbar. The control program reacts involving the entire module, closing the supply valve and opening the emptying valves. As the harmful condition has been recovered the Go_Manual command can be issued.

For each elementary unit, the FSM communicates to the field devices through a software interface implemented in Instruction List language (IEC 61131-3 standard). This latter reads the raw values from the sensors, it verifies the correct operation and converts the acquired data in engineering appropriate values according to the specific calibration curve. Furthermore, this interface receives the actions from the FSM block and converts them into the appropriate state value of the actuators. This approach guarantees the insulation of the FSM to the physical characteristics of the field devices and it eases the maintenance operations.

Supervisory Level

The supervisory level is the CS upper level. It supervises, via ALICE LAN Ethernet, the control process through a PVSS II as Supervisory Control And Data Acquisition (SCADA) system [11]. This level enables the interfacing and the integration in HMPID DCS. It provides the human-machine interfaces, alarm message handling and allows the data recording. The database is implemented as the ORACLE Real Application Cluster (RAC) [12]. All data stored in the archive are available for display and analysis using a graphical User Interfaces (UI).

The communication between the PLCs and PVSS has been established via the Siemens S7 driver [13] which is able to poll and write data to a Siemens S7 PLC.

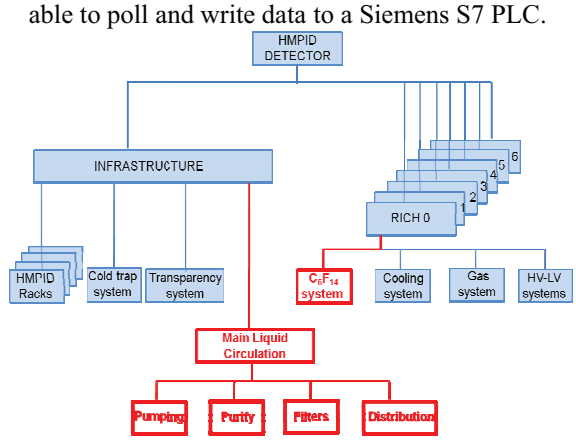


Figure 6: HMPID DCS control hierarchical structure.

Figure 6 shows the control hierarchical tree structure of HMPID DCS. It is type detector oriented hierarchy where HMPID is the top-node and 8 branches exist at the first level:

- seven for controlling the 7 HMPID modules. Each one hosts the control of the HMPID ancillary systems;
- one, called "infrastructure", dedicated to control all the other subsystem

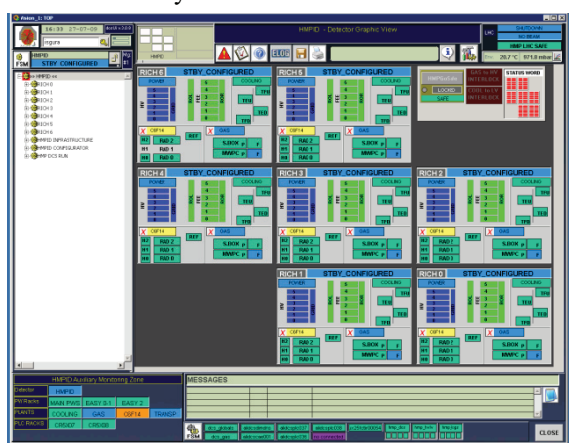


Figure 7: HMPID DCS UI.

Except for LCS, the FSMs of the other nodes are implemented in PVSS using the toolkit provided by the JCOP Framework and based on State management interface (SMI++)[14, 15]. The FSM browsing tree to navigate through the detector logic structure, relevant monitoring panels, and control elements is integrated in the standard UI developed for HMPID (Figure 7).

Figure 8 shows the UI RICH 0 liquid system (as seen in the FSM browser on the left side). As soon, CS registered a R_2 pressure value lower than the predefined threshold (red square in box 1), went immediately into a "FATAL UNDER" state, being the dominant alarm condition (as seen in box 1).

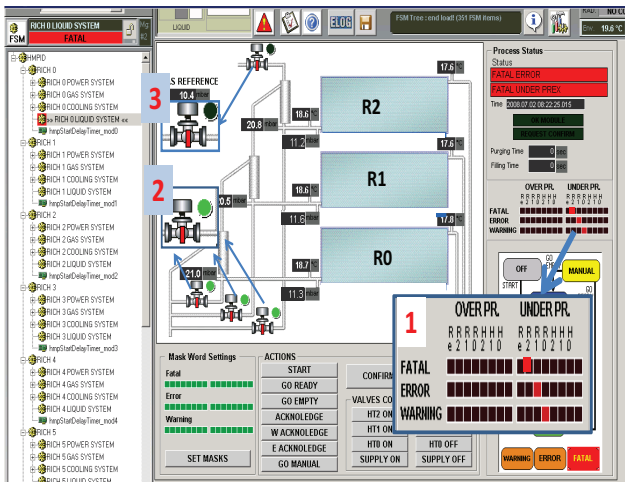


Figure 8: $RICH_0$ liquid system UI when in FATAL state.

It operated closing the emptying valves which are "normally open" (box 2) and the supply valve which is "normally closed" (box 3). At the same time HMPID DCS SMS system tool was enabled to send an alarm message to the expert operator via the GSM network mobile. When the harmful condition has been recovered the Go_Manual command can be issued.

CONCLUSIONS

C_6F_{14} CS was conceived and implemented according to the industrial standards using the Siemens S7 300 PLC as control devices, FSM structure for modular and automatic command execution and PVSS as SCADA environment.

CS allows LCS to run for 24 hours a day, correctly and safely. It is flexible, user friendly both for expert and no expert operator, coherent and homogenous with all the detector control sub-systems and it represents a cost-effective solution.

Moreover, CS showed an excellent capability to cope with harmful conditions that were not foreseen in the original LCS design.

Since the first LHC operation in 2008, LCS under CS control has enabled HMPID detector to successfully collect data.

Acknowledgements

We would like to thank A. Dell'Olio, M. Dell'Olio, Y. Leseneschal, J. Van Beelen and V. Rizzi for their fundamental contributions to the installation and cabling of the C_6F_{14} recirculation system and Control System. We would like also to thank Bosteels for the technical support.

REFERENCES

[1] "ALICE HMPID Tech. Des. Rep.", CERN/LHCC 98/19.

[2] F. Piuz et al., Nucl. Instrum. Meth., vol. A433, p. 222, 1999.

[3] C. Pastore et al. "The Cherenkov radiator system of the high momentum particle identification detector of the ALICE experiment at CERN-LHC". Nucl. Instrum. Meth., vol. A Physics Research A 639 (2011) 231–233.

[4] I. Sgura et al. "Sensors for monitoring water and oxygen in the VUV spectrometer of ALICE-HMPID detector at CERN-LHC". IEEE Catalog Number: CFP11IWI-USB ISBN: 978-1-4577-0622-6.

[5] G.Volpe, "Test results of the ALICE-HMPID detector commissioning" WSPC–Proceedings October 2007 Villa Olmo, Como

[6] G. De Cataldo, et al. "The ALICE-HMPID Detector Control System: Its evolution towards an expert and adaptive system". Nucl. Instrum. Meth., vol. A 639 (2011) 211–214.

[7] Profibus- Profibus & Profinet, June 2004 http: www.profibus.com.

[8] Berger, "Automating with SIMATIC: Integrated Automation with SIMATIC S7-300/400: Controllers, Software, Programming, Data Communication, Operator Control and Process Monitoring" 3.

[9] http://www.automation.siemens.com/mcms/simaticcontroller-software/en/step7/simatic-s7graph/Pages/Default.aspx

[10] http://itcobe.web.cern.ch/itcobe/Projects/Framework/Documentation/guidelinesDocument.pdf.

[11] http://itcobe.web.cern.ch/itcobe/Services/Pvss.

[12] P.Chochula, et al. "Handling large amounts of data in ALICE", ICALEPCS07, Knoxville, USA 2007.

[13] http://ab-project-unicos.web.cern.ch/ab-project-unicos/S7 Unicos/docs/NativeDriver_S7.pdf

[14] http://itcobe.web.cern.ch/itcobe/Projects/Framework/welcome.html/.

[15] B. Franek and C. Gaspar, "SMI++ - an object oriented Framework for designing distributed control systems", IEEE Trans. Nucl. Sci., Vol 45, Num 4, August 1998, pp.1946-1950.

TANGO INTEGRATION OF A SIMATIC WINCC OPEN ARCHITECTURE SCADA SYSTEM AT ANKA

T. Spangenberg*, K. Cerff, W. Mexner, KIT, Karlsruhe, Germany
Volker Kaiser, Softwareschneiderei GmbH, Karlsruhe, Germany

Abstract

The WinCC OA [1] supervisory control and data acquisition (SCADA) system (previous PVSS II) provides at the ANKA synchrotron facility a powerful and very scalable tool to manage the enormous variety of technical equipment relevant for house keeping, beamline, and machine operation. Crucial to the applicability of a SCADA system for the ANKA synchrotron are the provided options to integrate it into other control concepts even if they are working e.g. on different time scales, managing concepts, and control standards.

Especially these latter aspects result into different approaches for controlling concepts for technical services, storage ring, and beamlines.

The beamline control at ANKA was originally fully based on SPEC and is currently moved to TANGO [2] and SPEC [3] which has been fully integrated by expanding by TANGO server capabilities. This approach implies the essential need to provide a stable and fast link, which does not increase the dead time of a measurement to the slower WinCC OA SCADA system. The open architecture of WinCC OA offers a smooth integration in both directions and therefore gives options to combine potential advantages, e.g. native hardware drivers or convenient graphical skills.

The implemented solution will be presented and discussed at selected examples.

CHALLENGES

An important factor for the scientific excellence of a synchrotron is based on its beamlines and their rapid and precise measurement instrumentation and data collection. Obviously a huge number of specialised devices is to be managed. TANGO turned out to be an appropriate tool regarding the speed and the required flexibility for the experimental instrumentation. At the same time and time scale essential environmental information needs to be provided by the SCADA system.

The initially much slower SCADA provides at smallest engineering cost level the required long term stability and scalability and guarantees the robust control of PLCs and any house keeping equipment. But the different event execution times of WinCC OA in respect to TANGO requires an adoption of the data processing.

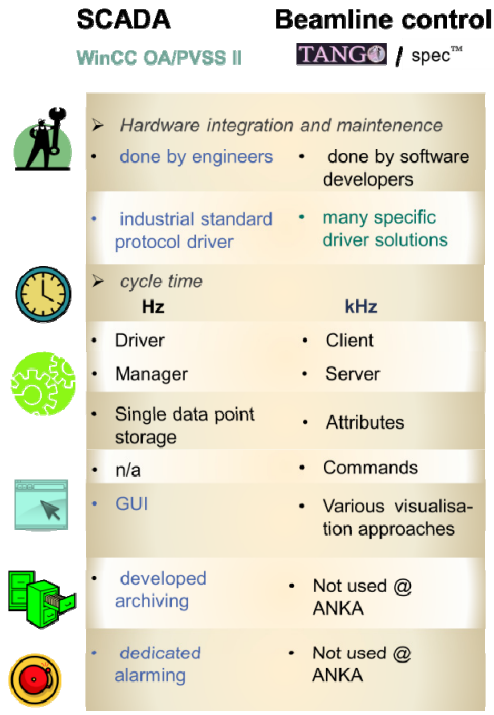


Figure 1: Comparison chart of basic features of WinCC OA and Tango based beamline control. Key benefits of the SCADA and TANGO used at ANKA are marked blue.

Both control systems show a certain analogy but at the same time different fundamental approaches. WinCC OA is set up on central event manager(s) which guarantee the consistency of the data, generate events, derived process values, and a bunch of other management functions. In contrast TANGO offers a clear client-server-concept. It results into a flexible and fast control structure which doesn't stand out ab inito as a clear interaction scheme or presentation concept.

Due to the fact that none of both systems may substitute the advantages of the other one a seamless cooperation needs to be sought.

SOLUTION

Both systems set up on TCP/IP communication and offer a complete API for Windows and Linux. The manager/driver concept of WinCC OA is comparable to TANGO server/client pair. Hereon set up the approach to combine manager – server and driver – client pairs.

At ANKA Linux was chosen as the main platform and all mangers have been designed to this environment. But there are no principle limitations to that operating system.

*thomas.spangenberg@kit.edu

The WinCC OA driver is in relation to TANGO a client which connects by multiple standard TANGO proxies to the servers. Polling and event based data point connections may be realised by the driver. The productive version at ANKA is based on polling. Some minor data type adaptions are done on driver level.

The TANGO server implementation for WinCC OA permits a straight forward read and write access to the WinCC OA data set. Supported data types of the current version are, integer, boolean, double, and string.

A WinCC OA data point and its addressed element are well described by its name which is a unique string and it is used as an index for the cache. Type and name are used to connect to the WinCC OA event manager and the requested value will be cached. Any change of it will update the cache.

Any update of a value is done in the comparable slow WinCC OA time scale. But a request to a successfully connected data point is handled by the TANGO server interface and is defined by the speed of C++ map. The response time of these maps is considerable shorter than 1ms and scales logarithmically to the size of these objects. In case of a higher workload the application is free scalable via the number of the server applications. Additionally the load to the WinCC OA event manager by multiple access of the same value from different clients is reduced by a cache mechanism, which effectively prevents the single threaded WinCC OA event manager to be blocked by an excessive number of client requests.

The write access to WinCC OA is indirectly feeding an input queue to the event manager. Again, timing issues needs to be considered. Requests from TANGO clients are queued and handled sequential with WinCC OA manager speed.

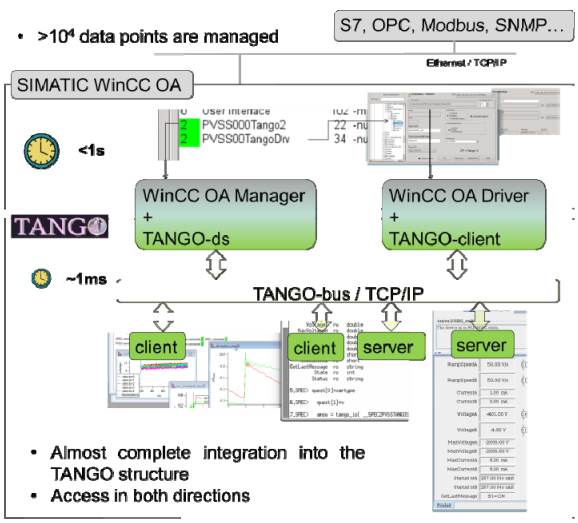


Figure 2: Schematic overview of the WinCC OA / TANGO interaction for beamline operation. The intrinsic time scale difference is managed by the developed WinCC OA manager and driver.

TECHNICAL CHALLENGES

Both systems are offering a complete C++ API which compiles each separately with gcc 4.x compilers. But the standard makefiles claim different compiler options and libraries. Additionally the WinCC OA is limited to 32bit.

Due to incompatible header requirements a direct exchange of data types or classes doesn't work. Dedicated data exchange classes were developed which are connecting the WinCC OA and TANGO part by a bidirectional data queue and granting the thread safe access to the exchanged data. There are different approaches for the TANGO server and for the TANGO client.

A major problem of combining a TANGO server and a WinCC OA manager is that both are claiming to define the int main() originally. Even if the source is available and the name conflict might be resolved it turned out that at first the TANGO server needs to start up and the WinCC OA manager template part is possible to run in a separate thread with a renamed and modified main() function. As a third argument the address of the data exchange is passed for a separately compiled library which contains the complete WinCC OA manager.

For compatibility reasons of the by POGO generated TANGO server the WinCC OA part parameters had to be defined as TANGO server property and are forwarded to the WinCC OA thread. To handle the TANGO server as a 'normal' WinCC OA manager, it was wrapped by a script which overcomes the difficulty of setting natively unknown command line TANGO parameters to a WinCC OA application by its management tools.

The combination of TANGO clients and WinCC OA drivers shows smaller difficulties. The program was developed as a standard WinCC OA driver project. The TANGO client properties are provided by the standard TANGO class and operated in an own thread. The same queue classes of the manager are used to create a thread safe data flow. Due to its original WinCC-OA nature the driver is also managed by the WinCC OA management.

ACTUAL STATE

The server (manager) as well as the client (driver) are in production state and hosted on a VMware ESX server as a small 32bit SuSE Linux 1GB RAM virtual machine.

The server gives a response of less than 1ms request to a request of a previously 'connected' value. Scalars of the type bool, int, double, and string are implemented in the current version. Other types will be implemented on request.

The client operates as a native WinCC OA TANGO driver and permits a standard WinCC OA data integration with the only constraint that it is currently not possible to browse the existing tango attributes of a device server.

CONCLUSION

The combination of the SCADA system WinCC OA and TANGO is fruitful for both sides. The TANGO world gains a well-engineered alarming and logging and a bridge to industrial protocols. Many industrial devices are now available to TANGO without any extra driver development. Due to caching any WinCC OA data can be accessed without any time loss in the timescale of the typical TANGO measurement processes. On the other side WinCC OA application range with its easy and straight forward to develop GUI is significantly extended for scientific applications by the huge number of already developed TANGO device servers and it's attributes and simple commands. And last but not least the WinCC OA TANGO driver allows automation engineers a seamless access to TANGO servers without special TANGO knowledge and general software development skills. The load of GUI development for standard automation tasks is shifted from software developers to engineers and therefore reduces the overall engineering costs and frees resources for more complex software tasks.

The solution is in productive state at ANKA.

REFERENCES

[1] http://www.etm.at
[2] http://www.tango-controls.org
[3] http://www.certif.com

UNICOS CPC NEW DOMAINS OF APPLICATION: VACUUM AND COOLING & VENTILATION

D.Willeman, E. Blanco, B. Bradu, J. Ortola, CERN, Geneva, Switzerland

Abstract

The UNICOS (**UN**ified **I**ndustrial **CO**ntrol **S**ystem) framework, and concretely the CPC (Continuous Process Control) package, has been extensively used in the domain of continuous processes (e.g. cryogenics, gas flows) and also in others specific to the LHC machine as the collimators environmental measurements interlock system. The application of the UNICOS-CPC to other kind of processes: vacuum and the cooling and ventilation cases are depicted here.

One of the major challenges was to figure out whether the model and devices created so far were also adapted for other type of processes (e.g. Vacuum). To illustrate this challenge two domain use cases will be shown: ISOLDE vacuum control system and the RFQ4 and STP18 (cooling & ventilation) control systems. Both scenarios will be illustrated emphasizing the adaptability of the UNICOS CPC package to create those applications and highlighting the discovered needed features to include in a future version of the UNICOS CPC package.

This paper will also introduce the mechanisms used to optimize the commissioning time, the so-called virtual commissioning. In most of the cases, either the process is not yet accessible or the process is critical and its availability is then reduced, therefore a model of the process is used to validate offline the designed control system.

UNICOS-CPC BASICS

UNICOS (UNified Industrial COntrol System) with its CPC (Continuous Process Control) package is a control framework which was created to design, develop and finally deploy control systems applied mostly to industrial continuous processes [1].

UNICOS-CPC intervenes in the three classical layers of an industrial control system although provides deliverables only for the control and supervision layers. (1) At the field layer, actuators (e.g. valves, pumps...) and sensors (e.g. temperature or pressure measurements) are extensively described in the UNICOS-CPC control system specifications. (2) At the control layer, the framework provides both, a PLC (Programmable Logic Controller) baseline with basic objects and methodology to design the control system.

After an UNICOS unit-oriented design, the process control architecture is defined. A basic structure containing specific devices (e.g. process control units, field objects…) and the interactions between them will be automatically generated. In most of the cases the user introduces the process specific control logic in predefined placeholders keeping the consistency and homogeneity of different applications. (3) At the supervision layer, the so-called SCADA (Supervisory Control and Data Acquisition) layer, the UNICOS-CPC framework will provide also a baseline with all the devices mentioned before and the peer instances already created for the control layer.

Populating those instances, which are automatically generated for the layers of control and supervision, naturally avoids the synchronization problems between them. In addition, the communication resources for the PLCs and the SCADA are also provided and deployed in both layers. Only the control logic, which is process dependent, has to be coded by the control developer.

All plant actuators and sensors are modeled as one or combined in several UNICOS types. The idea of UNICOS-CPC has always been to keep the devices as generic as possible willing to homogenize both design and operation. As a practical example, motors, pumps and digital valves are modeled with a basic type called *OnOff*.

Other UNICOS-CPC strength is the plant operator capability to intervene any time and any place in the plant through its control components. To achieve this, all field objects (e.g. valves, pumps, heaters…) never have direct access to the I/O of the PLCs but through appropriate I/O types. A general view of the UNICOS-CPC objects can be seen in Fig. 1. There are four types of objects: (1) I/O objects (e.g. Analog Input) which link the sensors cabled on a particular channel to the control system, (2) the plant actuators are then modeled by field objects (e.g. OnOff for discrete and Analog for analog actuators). (3) The control objects (e.g. Process Control Object) drive a group of field objects. The controller object (PID) is used for feedback control, and finally the (4) the interface objects used to parameterize alarm thresholds or showing basic plant status.

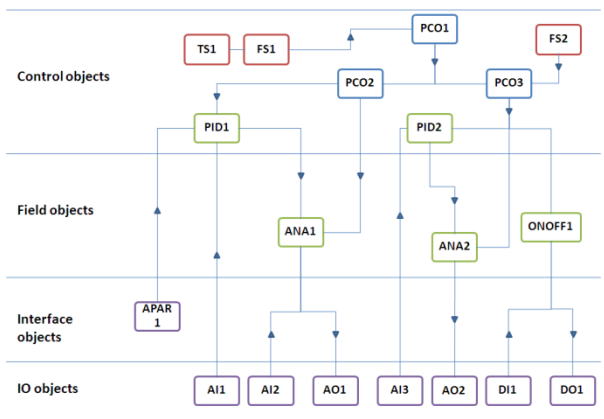


Figure 1: UNICOS-CPC objects hierarchy.

The advantage of a high granularity of types, which gives the framework a great versatility to combine objects to model new needs, could also be judged as an overload in new domain devices where an integration to create more complex devices is likely advisable.

The UNICOS framework was born in 1998 and was firstly used in the LHC cryogenic system. After some evolution and application to other processes (gas flows, interlocks…), we face another challenge: validate the approach in other different domains close to the continuous processes but still with additional requirements.

When CERN decided to replace the out-phased vacuum control system in ISOLDE, we had to check the validity of the UNICOS-CPC in a completely different domain: the vacuum. ISOLDE is a physics facility dedicated to the production of a large variety of radioactive ion beams for a large number of experiments. The vacuum was essential and a high critical component of the installation.

A new challenge started when CERN decided that the control systems of all CERN cooling and ventilation plants had to be done using the UNICOS-CPC framework. More than 150 installations, new and existing, will be deployed using the framework. A brief report of the first installations is provided in this paper.

UNICOS-CPC APPLIED TO VACUUM CONTROL

In large physic experiments, vacuum systems are vital to keep a beam of particles alive and also compulsory in employing cryogenic systems and superconducting magnets. Vacuum can, depending on the experiment, be more or less critical and then the choice of actuators and sensors employed will highly depend on these needs. Other characteristics of the vacuum process are the high repeatability of the control system together with the absence of feedback regulation and predominance of discrete systems.

Vacuum systems are mostly composed by a set of different kind of pumps (e.g. primary, turbomolecular...), and different kind of pneumatic or electrical valves. The measurements are mostly focused on pressure sensors with different ranges of validity.

Most of these devices are usually found in different domains, concretely in the continuous process industry. This is one of the reasons why we believed that the UNICOS-CPC framework was adequate to vacuum installations. The existing UNICOS-CPC types library easily fitted the vacuum requirements.

The ISOLDE vacuum control system was the first vacuum system built with the UNICOS-CPC framework [2]. The vacuum of ISOLDE contains 19 vacuum sectors, 13 pumping stations, 2 venting systems and 1 exhaust facility; see the Fig. 2 where a plant schema is represented. All these subsystems are composed of the devices depicted in Table 1.

Table1: ISOLDE Equipment

Equipment	Number
Valves	180
Turbomolecular pumps	30
Primary pumps	20
Pirani gauges	80
Penning gauges	20

All the penning and Pirani gauges are connected to TPG300 controllers. These controllers are then connected to the PLC via Profibus, an industrial fieldbus.

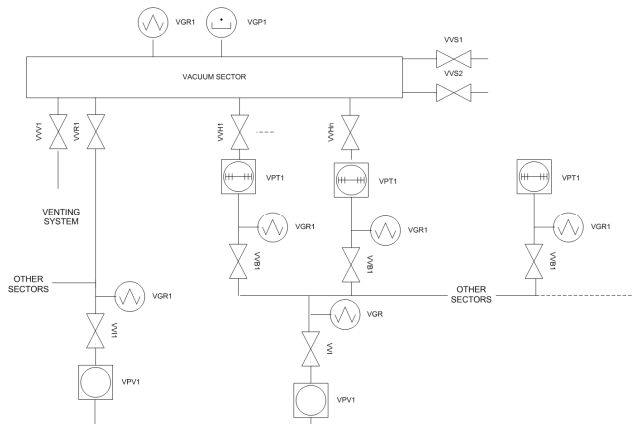


Figure 2: Plant schema of an ISOLDE vacuum sector.

This type of process is highly repeatable. All the vacuum sectors are very similar in ISOLDE and only the number of turbomolecular pumps and its attached valves varies. The goal with installations of this type is to create a set of templates to generate automatically the associated logic only depending in a parameterization. (See Fig. 3). These parameters are the only interaction with the process expert. Once that information is in the UNICOS-CPC spreadsheet specification, the control system logic is automatically generated without a single line of code typed by the developer. This approach considerably saved time for the control system development and for the subsequent upgrades.

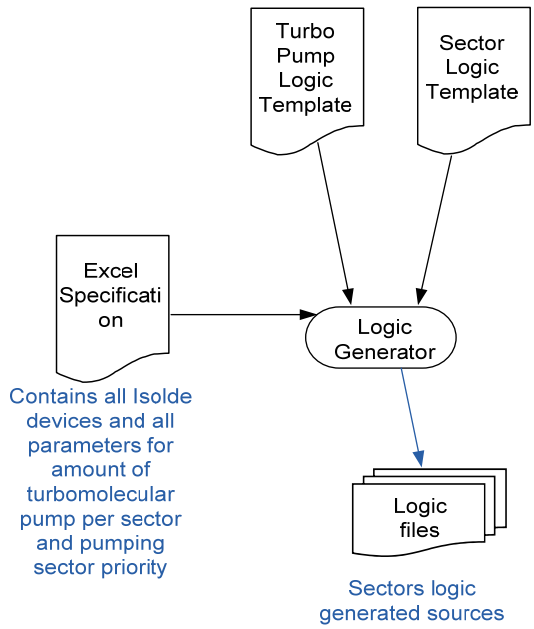


Figure 3: Principle of automatic logic generation.

Following the same philosophy, the operation synoptics were described as templates and with only a few parameters introduced, all the supervision synoptics were automatically created. Again a precious time saved for both, development and maintenance.

The ISOLDE control requirements were challenging. Pumping sector priority management, possibility to manage vacuum level sector by sector and the criticality of the exhaust facility to avoid an undesired release of the irradiated gases contained in ISOLDE, were easily managed with UNICOS-CPC by only parameterization in the control specification.

As an example of matching vacuum devices to UNICOS-CPC objects, the case of the TPG300 vacuum device measuring the pressure combining Pirani and Penning gauges could be analyzed. The final solution combined several basic UNICOS-CPC types to keep the minimum functionality required: only the analogue values and statuses were extracted from the devices and placed in appropriate based UNICOS-CPC objects.

The extraction of all of the information coming from such devices will end up in combining several I/O and interface basic objects. This was one of the motivations to make evolve the UNICOS-CPC framework: to cope with the need of being more flexible in the creation of new types when required. The new UNICOS-CPC v6 package provides the tools to create easily new types in a standard way [3]. This increases the flexibility of UNICOS-CPC to produce new control functionalities and to fulfil operators and process engineers requirements.

UNICOS APPLIED TO COOLING & VENTILATION CONTROL SYSTEMS

Cooling and Ventilation (CV) systems embed very large types of processes. Cooling systems allow the production of cold water by means of chillers or refrigeration towers. They are mainly composed by industrial devices like valves, heat exchangers, compressors, pumps and fans. This kind of processes can be classified as continuous. Then, the HVAC (Heating, Ventilation and Air Conditioning) are performed by air handling units ensuring the control of the indoor air quality (temperature, humidity, oxygen, cleanness). In the scope of accelerators and large experimental physics, the ventilation must be ensured inside tunnels, experimental caverns, experimental halls or computing centers. HVAC systems are also composed by industrial equipments (e.g. heat exchangers, heaters, fans, dampers, valves, filters…).

The control of a refrigeration cooling plant, called STP18 (See Fig. 4), and the cooling of the radiofrequency quadrupole for the future CERN LINAC (RFQ4) were deployed using the UNICOS-CPC package. Once more the framework showed its nature of adaptation to new process domains.

Concerning air handling units, the first control projects are starting up using UNICOS-CPC. Due to the nature of such processes and to handle the new requirements imposed by HVAC systems, the UNICOS-CPC basic package has been enriched with new operation widgets (equipment representation) in a first stage.

The UNICOS-CPC v6 includes the needs in terms of new types and requirements coming from its application to this new domain. These include a new library of HVAC widgets, a color library to show flows of different fluids (e.g. water, gas,…) and additional features such a more secured interlock handling system on some field equipments and automatic integration of regulation structures (e.g.: cascade control).

Moreover, due to different operation needs in Cooling and HVAC systems, where control rooms are not necessarily present and where operation is done via industrial local panels, UNICOS-CPC has been embedded in industrial touch panels in order to ensure local operation at any time.

In the next years, most of Cooling and Ventilation systems at CERN will be progressively upgraded using the UNICOS-CPC framework.

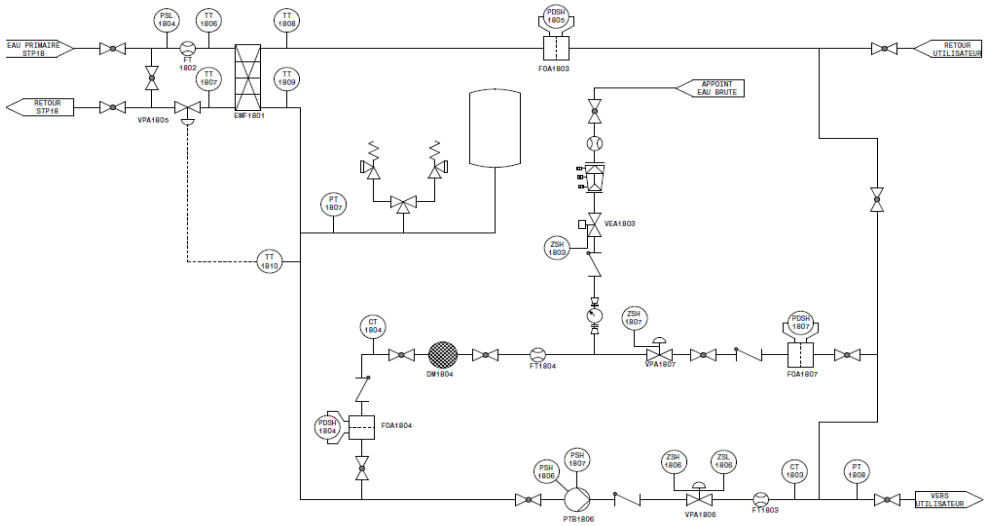


Figure 4: Plant schema of the STP18 cooling plant.

VIRTUAL COMMISSIONING

A cryogenic process simulator able to be plugged to UNICOS-CPC control system has been developed in the last years [4]. This simulator reproduces the process dynamic behaviours based on a set of DAE (Differential Algebraic Equations) describing the physics of the processes. The model of the process is connected to the real control system, reading the PLC outputs (actuators) and simulating the PLC inputs (sensors) in real-time.

The setup of such a simulator allows the development of virtual commissioning platforms in order to make a commissioning of the control system (PLC program and supervision) in advance. Virtual commissioning is a very powerful tool in case of control system upgrades when the commissioning time has to be reduced to minimize the shutdown time of a critical process. In addition to this, the virtual commissioning allows a faster and safer check of the PLC control program where all functional sequences can be tested with the simulator without any risk.

The current process simulator used at CERN for cryogenics has been extended and also supports water cooling simulations. The two systems STP18 and RFQ4 were successfully simulated ensuring an efficient commissioning. Due to the critical nature of some ventilation installations, a dedicated library for HVAC systems will be developed.

CONCLUSION

The ISOLDE control application for vacuum systems, RFQ4 and STP18 for cooling systems were successfully implemented using UNICOS-CPC framework. For all projects, the UNICOS principles applied during the operation and the SCADA features were quickly understood by the operators. The UNICOS-CPC application architecture has been assimilated by vacuum control experts, and since the first implementation, they are autonomous and able to develop and maintain their own control projects. This generic way of control system programming shows its high versatility and its easy learning in different domains of engineering.

The UNICOS-CPC v6 framework will cover missing functionalities identified during the development of both, vacuum and HVAC, control systems. Moreover, the new framework reinforces the local operation by using appropriate industrial devices. It also provides new tools able to create new types and a generic interface to perform virtual commissioning using a simulator.

REFERENCES

[1] Ph. Gayet and R. Barillere “UNICOS a framework to build industry like control systems”, ICALEPCS’05, Geneva, October 2005.

[2] Blanchard, S. et al. “The New Control System for the Vacuum of ISOLDE at CERN”, ICALEPCS’11, Grenoble, October 2011.

[3] E. Blanco et al. “UNICOS CPC v6: evolution”, ICALEPCS’11, Grenoble, October 2011.

[4] Ph. Gayet and B. Bradu “PROCOS: a real time simulator coupled to the control system”, ICALEPCS’09, Kobe, October 2009.

SUPERVISION APPLICATION FOR THE NEW POWER SUPPLY OF THE CERN PS (POPS)

H. Milcent, X. Genillon, M. Gonzalez-Berges, A. Voitier, CERN, Geneva, Switzerland.

Abstract

The power supply system for the magnets of the CERN PS has been recently upgraded to a new system called POPS (POwer for PS). The old mechanical machine has been replaced by a system based on capacitors. The equipment as well as the low level controls have been provided by an external company (CONVERTEAM). The supervision application has been developed at CERN reusing the technologies and tools used for the LHC Accelerator and Experiments (UNICOS and JCOP frameworks, SIMATIC WinCC Open Architecture SCADA tool). The paper describes the full architecture of the control application, and the challenges faced for the integration with an outsourced system. The benefits of reusing the CERN industrial control frameworks and the required adaptations will be discussed. Finally, the initial operational experience will be presented.

THE NEW POWER SUPPLY OF THE CERN PS

The Proton-Synchrotron (PS) accelerator at CERN is supplied by a new power system (see Fig. 1). This new power system called POPS (Power for PS) [1] replaces the existing power system (including a 90MVA rotating machine), which feeds the main magnets of the PS accelerator. The PS magnets require a maximum DC current of 6kA with a maximum DC voltage of ±10kV. The new power system is based on three-level converters with capacitive energy storages and is rated at 60MW peak power. POPS will supply the PS magnet without drawing all the power from the network. The maximum stored energy in the magnets is 14MJ. The capacitor banks supply this energy. DC/DC converters control this exchange of energy between the capacitor banks and the load. The system is connected to the electrical network by two AC/DC converters (Active Front End). These converters charge the capacitor banks and supply the losses of the load and of the converters.

The POPS control is made up of two different structures that interact to provide the required performances; these are the Function Generator/Controller (FGC) and the CONVERTEAM [2] (CVT) control system. The FGC generates the reference voltage (V_{ref}) for the POPS output voltage and the CVT control system is in charge of producing and maintaining it. The CVT control software has been developed in a proprietary environment called P80i. Once the V_{ref} is set by the FGC the CVT control has to divide it into as many components as the number of DC/DC converters connected in series. This operation called "Voltage Dispatching" is done by assigning coefficients proportional to the amount of energy stored in the capacitors (based on the maximum and minimum voltage for the capacitors that is different depending on whether the capacitor is of floating or charger type). As a general indication the voltage dispatching is done by allowing the floating capacitors to take only part of the inductive contribution to the V_{ref}, leaving all the rest to the chargers.

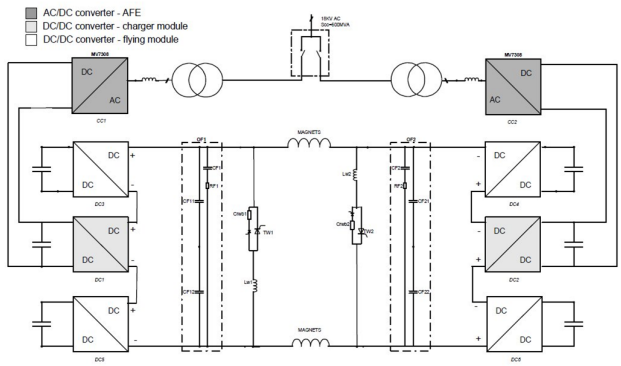


Figure 1: Topology of the power system.

CONTROL ARCHITECTURE

The POPS control system architecture (see Fig. 2) is a three layer control system composed of:

- Field layer with remote input/output module connected to the control layer. This layer is composed of WAGO modules interfaced with the control layer via the PROFIBUS fieldbus.
- Control layer with CONVERTEAM hardware and software based on VME cards, holding the closed control loops and the voltage and capacitors control. The CONVERTEAM control system is distributed over four CPUs and remote IO modules. Each CPU is monitoring the link with the Supervision layer; the interruption of the communication stops the powering of the system.
- Supervision layer. In this layer, process views are provided for the monitoring and the control/command interface of the control system. Other typical functions (e.g. archiving, trending, alarm handling) are also provided by this layer.

The application is usually supplied as a complete turnkey system by CONVERTEAM. However in the context of CERN, tools for the Supervision layer already exist, allowing a smooth integration into the CERN control complex. The Field and Control layer were provided by the company manufacturing the power system. The Supervision layer was developed at CERN. Furthermore, with the usage of CERN tools the interface to the FGCs and the integration of data from the cooling and ventilation subsystem via DIP (Data Interchange

Protocol) [4] was very straightforward. This information is made available in the POPS process view and it is very useful for the operation.

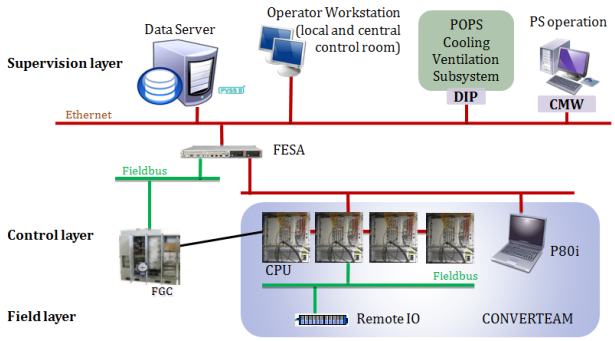


Figure 2: POPS control hardware architecture.

PUTTING IT ALL TOGETHER

Interfaces

Two solutions were used for the interface to the CONVERTEAM control layer: OPC (OLE for Process Control) [3] and a proprietary protocol based on TCP/IP. The first deployment of the supervision application was done with the OPC interface. Later a dedicated front-end program based on the FESA framework [5] was developed to interface with the proprietary protocol.

All the devices defined in the control layer are listed in an Excel spread sheet, which is used by CONVERTEAM to configure their system with the P80i programming console. A dedicated tool (see Fig 3) was developed to extract from this Excel spread sheet all the information for the supervision layer: list of devices, definition, configuration, etc.

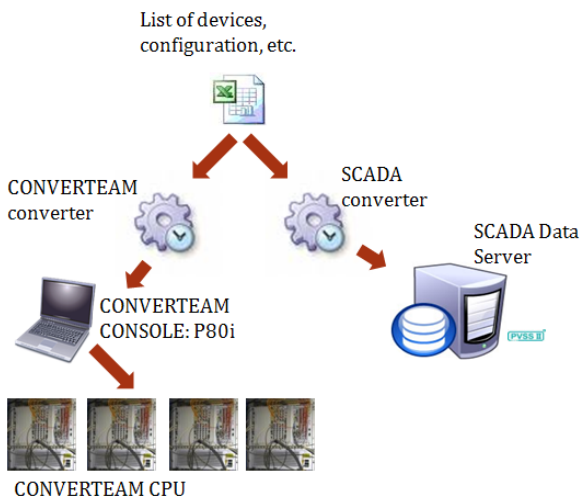


Figure 3: System configuration workflow.

SCADA Framework

The supervision layer of POPS was based on the UNICOS framework [6] developed at CERN. This framework is used to develop control applications. It provides the developers with the means to rapidly develop full control or monitoring applications and provides the operators with ways to interact with all the items of the process from the most simple (e.g. I/O channels) to the high level composite devices. In addition, UNICOS offers tools to diagnose problems in the process, the control system and to access and operate the devices without specific development. It is based on SIMATIC WinCC Open Architecture ® [7] (WinCC OA previously released under the name of PVSS) and it offers at the supervision level, in both Linux and Windows operating systems, functionalities such as: interface to include additional packages, device and file access control, interface to the LHC software suite, a homogenous user interface entirely customizable with features such as navigation capabilities between panels and trends (WWW browser like, contextual buttons, pop-up navigation), access to the devices without creating a specific view (Tree Device Overview), process alarm and event list. A device includes some libraries of code, a set of widgets (summarized view of the device status), a faceplate (detailed view of the device) and the device actions interface. A tool is provided to import the device configuration into the supervision layer. A developer of a UNICOS supervision application does not need any deep WinCC OA knowledge (e.g. scripting). He just needs to import the instances of the process devices; to create the application specific process views using the provided catalogue of devices and to configure the required trends.

Development

The use of the UNICOS framework allowed us to develop the supervision in two phases:

- Phase I: with low-level dedicated devices interfaced via OPC to the control layer. During this phase the equipment and the feedback control loops were validated.
- Phase II: using the FESA interface instead of OPC. High level device widgets encapsulating the low-level devices were added to the Phase I devices. These widgets were developed using the UNICOS widget interface in which a widget can be attached to a hardware device or to a generic device. These widgets were used directly by the operators and the control experts to create the process views.

Both the low-level and high-level device widgets present the device data and the communication link state to the CONVERTEAM CPU. Dedicated 32 bit devices for alarm, interlock and default status were created. Analog input and 32 bit status devices were re-used from other packages without any new development.

INITIAL OPERATIONAL EXPERIENCE

The supervision layer was successfully used during the commissioning and the operation of POPS. Thanks to the UNICOS flexible configuration, the graphical interface (see Fig 4) was customized and divided in three main areas:

- A header: showing the global state of the control system: communication alarm link, shortcut for fast access to the views, user login.
- A status area: presenting the most important state information of the equipment.
- A bottom area: for the most important commands, these commands are protected depending on which user is logged in.
- A central area: where the detailed views for the operators are displayed. About 35 different views were developed for the operation and diagnostic: detail view on the AC/DC, DC/DC, output connections, cooling, interlocks, etc. An expert having no deep knowledge of WinCC OA made the views with the high level widgets.

During the commissioning, each input/output was tested in order to check the complete control chain, from the sensor up to the supervision with the Tree Device Overview, a UNICOS built in feature. Later on, the POPS complete graphical interface was released and commissioned. The event list, another built in utility, was also used to understand the sequence of faults in case of the powering failure of POPS. This event list is based on a sequential list of 32 bit state words. The individual bits for the statuses, the alarms and default states are extracted and ordered by time. As the timestamp of these events comes from the control layer, one can diagnose the exact sequence of faults.

The navigation features provided by the UNICOS framework were also very useful to go directly from one view to another one without having to search within the list of all the views.

Multiple instance of the POPS supervision graphical interface can be started from many places, for instance from local control room, during commissioning or development time, from central control room for the daily operation. This interface in the central control room is very useful for the operators to make a first diagnostic before calling the experts.

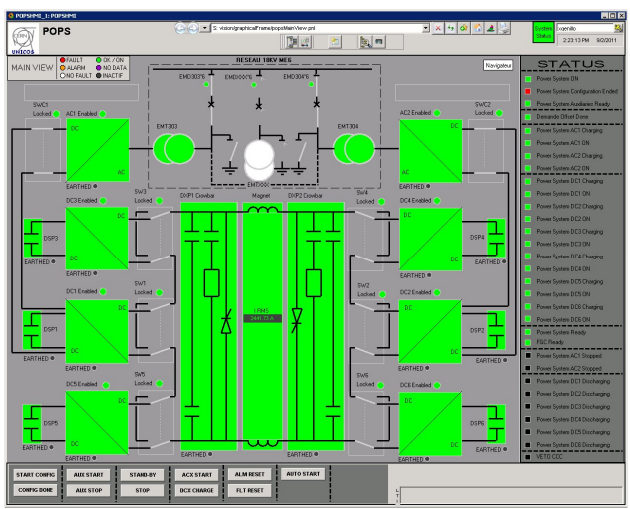


Figure 4: POPS graphical interface.

Later a dedicated interface to the PS operation tool was provided by using the server middleware interface [8] of the LHC Software suite built in UNICOS. This interface publishes the states of devices and a set of commands to act directly on the control layer: stop, start, etc. New states and commands can be easily added by reconfiguring this interface without any development.

CONCLUSION

The control systems of the LHC and the experiments have been running successfully for the last years. This success goes beyond the original scope, the same tools which had been used for their development were used to build new applications like POPS.

The reuse has only been possible because at an early stage of the different LHC projects it was decided to base the developments for industrial controls in a common generic frameworks and components. This choice implied a certain overhead which today is paying off when facing new projects.

The controls for POPS have been developed in a short time frame and with a small team. In addition, it has seamlessly integrated in the CERN controls infrastructure. Any future maintenance and upgrades will also be simplified.

REFERENCES

[1] Jean-Paul Burnet, CERN: New PS Power System POPS, 2nd Workshop on Power Converters for Particle Accelerators (POCPA), Geneva, CH
[2] http://www.converteam.com/
[3] OPC Foundation (www.opcfoundation.org/).
[4] DIP (Data Interchange Protocol): http://cern.ch/dip
[5] M. Arruat et al., "Front-End Software Architecture", ICALEPCS 2007, Knoxville, USA
[6] H. Milcent et al, "UNICOS: AN OPEN FRAMEWORK", ICALEPCS2009, Kobe, Japan.
[7] SIMATIC WinCC Open Architecture: http://www.etm.at.
[8] CMW (Common Middleware).

THE ELBE CONTROL SYSTEM – 10 YEARS OF EXPERIENCE WITH COMMERCIAL CONTROL, SCADA AND DAQ ENVIRONMENTS

M. Justus, F. Herbrand, R. Jainsch, N. Kretzschmar, K.-W. Leege, P. Michel, A. Schamlott
Helmholtz-Zentrum Dresden-Rossendorf, Radiation Source ELBE, Germany

Abstract

The electron accelerator facility ELBE is the central experimental site of the Helmholtz-Zentrum Dresden-Rossendorf, Germany. Experiments with Bremsstrahlung started in 2001 and since that, through a series of expansions and modifications, ELBE has evolved to a 24/7 user facility running a total of seven secondary sources including two IR FELs. As its control system, ELBE uses WinCC on top of a networked PLC architecture. For data acquisition with high temporal resolution, PXI and PC based systems are in use, applying National Instruments hardware and LabVIEW application software. Machine protection systems are based on in-house built digital and analogue hardware. An overview of the system is given, along with an experience report on maintenance, reliability and efforts to keep track with ongoing IT, OS and security developments. Limits of application and new demands imposed by the forthcoming facility upgrade as a centre for high intensity beams (in conjunction with TW/PW femtosecond lasers) are discussed.

THE ELBE ACCELERATOR FACILITY

The electron accelerator ELBE is a multiple user facility (see fig. 1) that delivers an electron beam between 5.5 and 33 MeV (pc) with a broad range of temporal schemes, ranging from single sub-pC bunches to a 1 mA cw beams with 13 Mhz repetition rate. Electrons are produced by a 250 kV thermionic d.c. gun and accelerated by two cryomodules, each comprising two 9-cell TESLA-cavities at 1.3 GHz. Until September 2011, the electron beam was alternately delivered to a bremsstrahlung facility, two IR FELs (5…280 µm), a site for channelling X-rays, cell irradiation and detector analysis, a neutron time-of-flight source, a position source and a preliminary electron-laser interaction. Additionally, a well noted superconducting RF gun making progress towards delivering bunch charges of up to 1 nC with significantly decreased transverse emittance. Thus, ELBE is able to address a wide range of scientific disciplines and programs within the Helmholtz community and above.

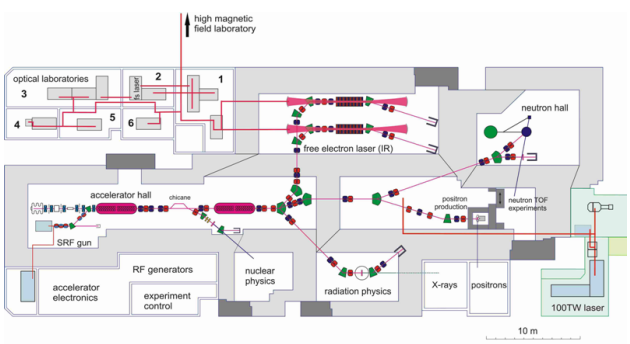


Figure 1: ELBE facility overview.

THE ELBE CONTROL SYSTEM

The ELBE project started in 1999, where the decision was made to use commercial industrial control technology as its control system (CS). Due to its project size, ELBE was sort of a pilot project in electron accelerator technology at the former Forschungszentrum Rossendorf. So this decision was in parts the result of balancing in-house expertise in industrial control systems versus the advantages of systems specifically developed for accelerators. As a multifunctional facility, ELBE has not been built all at once (or better: within few years), but rather grew proportionally with the installed beam lines and user labs.

Looking at the system architecture (fig. 3), the ELBE CS is a mostly straight-forward implementation of the automation pyramid as known from IEC 62264 [1] and depicted in fig. 2. On layers 0…2 we find different branches addressing specific needs of accelerator control: machine control and supervision, machine protection system and data acquisition. On layers 3 & 4 there are web based applications with no direct integration into the ELBE control system.

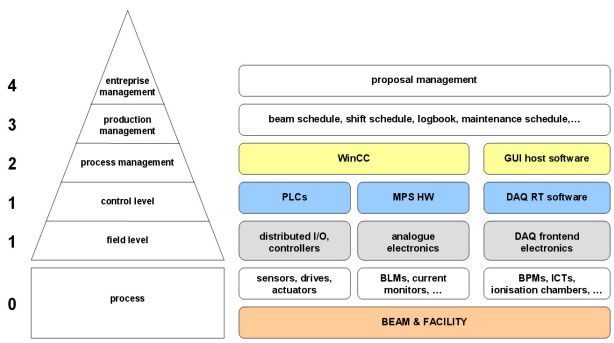


Figure 2: ELBE control system classification.

Machine Control and Supervision

At field level, we use distributed I/O systems interconnected by Profibus DP [2], such as Beckhoff field bus terminals [3], Simatic ET200 [2], or Profibus supporting field devices. “Intelligent slaves” are embedded here to source out specific control tasks from control level.

Control level is represented by Simatic PLCs [2], distinguished by their major areas of functionality, such as personal safety system (PSS), beam generation & beam

control, vacuum & media control, beam dump control and cryogenics. This separation was chosen to achieve an optimum match between tasks and technology of the central hardware and firmware. The software is developed in-house using IEC 61131 languages within the Step7 environment [2]. The CPUs are interconnected by Profibus communication to share main machine parameters. Most of them are within one single project.

For supervisory data acquisition and control (SCADA), we use WinCC, a Windows based system [2], in the form of a server-client architecture. Graphical user interface, alarm handling, data logging & trend display and management of machine settings are implemented. For specific systems, additional standalone WinCC stations or operator panels are available to ensure access independently from the main server. WinCC client stations communicate via LAN with the server.

DAQ Systems

Data acquisition for beam diagnostics, demanding fast triggering and high temporal resolution are realized with National Instruments PXI crates and PC DAQ components [10]. Application software programmed using LabVIEW or LabWindows™/CVI [4], running real time or Windows systems. They mainly use shared network variables for data exchange between host applications and acquisition systems. To interface PXI, PCs or user systems (NIM, VME) with the PLC world, we use OPC [5], specifically the Simatic Net OPC server [2], with defined access rights. Different data engines translate OPC item access into shared variables or telegram based TCP/IP communication, where OPC implementation is not available or inconvenient.

Machine Protection System

The MPS consists of a fast and a slow signal chain. The fast chain is realized with hardware electronics, including beam loss detection, vacuum inrush detection & prevention and the RF system interlock with a total reaction time below 2 milliseconds, using PLC interrupt techniques. Hard-wired data exchange with the main PLC enables altering the MPS functionality according to the current beam option, as well as a diagnostics interface. The slower branch of the MPS is spread all over the PLC code units, ensuring beam shutoff within 2 seconds due to technical failure or operational mistakes.

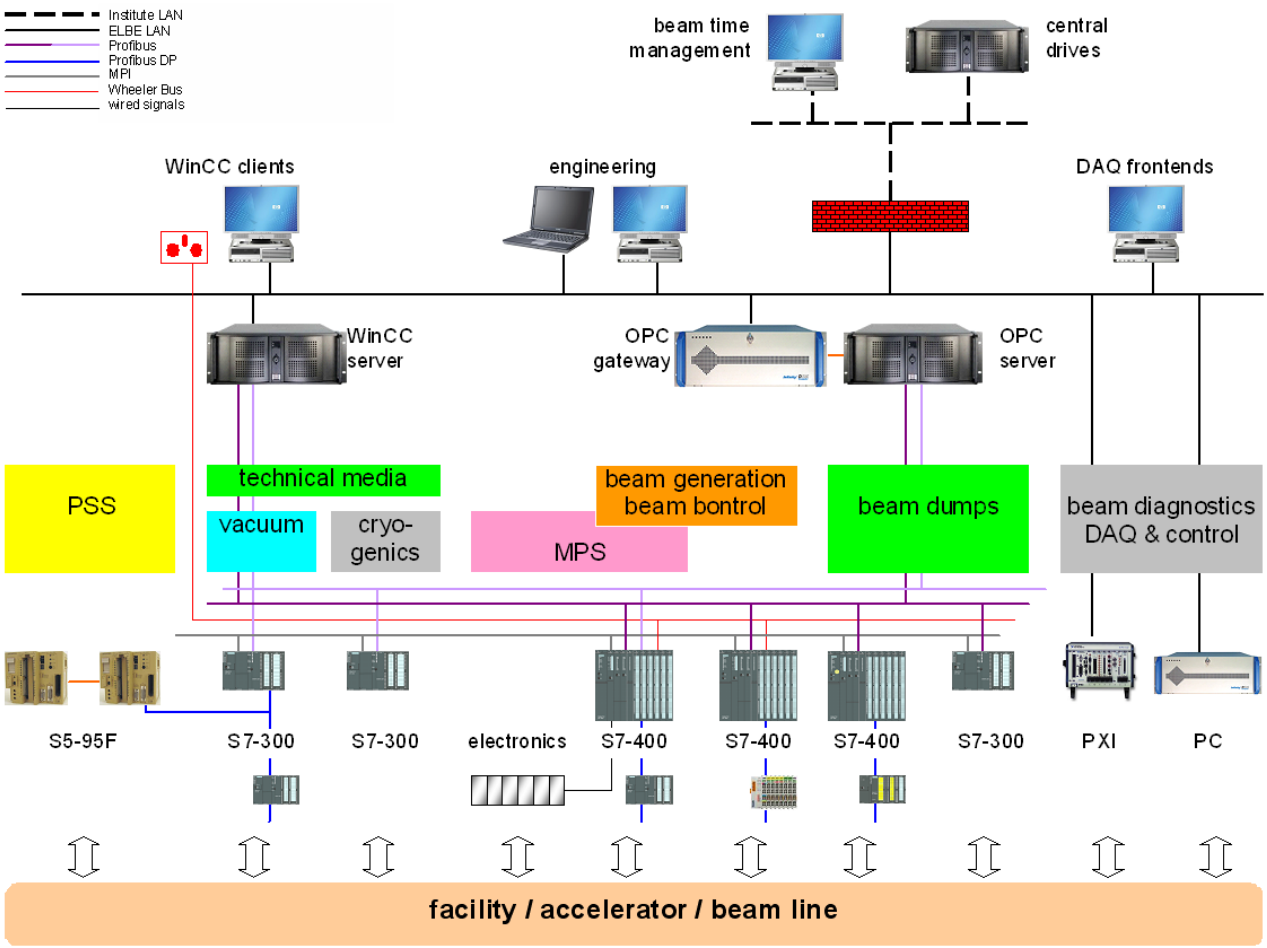


Figure 3: ELBE control system architecture.

Operational Management

ELBE operation is organized by a web based infrastructure like beam time schedule, shift schedules, a logbook with features like failure tracking and user response and a WIKI supporting the operator staff. The overlying administration of experimental proposals is a centralized service of the Helmholtz-Zentrum Dresden-Rossendorf. Integration of these tools into the machine control system is not yet available.

System Key Figures

As all pictures show only exemplarily the system characteristics, a few key number are given in the below table:

Table 1: ELBE Control System Key Figures (Rounded)

PLC CPUs	12
PLC data points	4500
PLC plain machine code	1.5 MB
PLC development software	Step7 V5.5
MPS fast channels	80
WinCC process tags	7000
WinCC client PCs	25
WinCC software	V7.0
OPC items	250
LabVIEW development software	2009
PC / PXI DAQ systems	10
PC / PXI DAQ applications	15
MPS fast hardware channels	80

OPERATIONAL EXPERIENCE AND SYSTEM MANAGEMENT

System Availability and Stability

As a measure for system availability, one can simply monitor the failure or unavailability rate of servers, PLC units, communications and other systems. As ELBE is a single beam linear machine, at least downstream the accelerator subsystems are not permanently needed to deliver beam. In fact, it is a special advantage that for beam time statistics, a critical failure scenario can often be compensated by changing the schedule. Thus, availability is more a result of managing infrastructure, system maintenance and implementing a proper failure monitoring to be one step beyond.

Hardware Management

Usual failure scenarios with few recurrences a year involving CS components are:

- PLC hardware failure, really rare if operational conditions are maintained as specified (this is in terms of radiation exposure not always possible)
- field bus components failure, can occur under hard EMC conditions
- PC & PXI system hardware failure, likely for standard PCs

To meet hardware and communication difficulties, a monitoring system has been established during the last three years to permanently check hardware, field devices (i.e. magnet power supplies, drives), field bus communication, serial connections, as well as CPU operation by means of PLC alarm handling and software diagnostics. Malfunctions are reported to WinCC, enabling statistical analysis within its alarm system. This often helped to detect sub system failure well before the specific components are needed for beam delivery. Commercial PLC technology has been experienced as ideal for implementation of simpler control functionality with a high demand on availability and physical robustness. The advantage of commercial hardware in general is that staff does not have worry about electronics design, availability of parts or firmware compliance, if good manufacturer support is provided.

Software Management

More often, due to permanent modification of the facility during the last years, we experience several types of software problems:

- programming errors in PLC code, WinCC scripts and other software that appear in unlikely and unforeseen system states
- WinCC server failures, i.e. due to project modification during system runtime or communication errors
- communication problems between DAQ components and applications
- malfunction of application software

Dealing with these problems requires (beyond the pure code verification) software and IT management measures. To become independent from overall IT policies (which are usually forced on any system that connects to the general IT infrastructure), the ELBE control system has been segregated from the overall IT network by an own domain behind a firewall (see fig. 3). Dedicated access rights and trusted relations for accounts and groups enable access to central services and drives.

For NI or Siemens application and development software, compatibility of operating systems, development software, hardware components and IT policies has to be checked carefully. Proven and tested configurations are made in-house standards. WinCC itself already offers log files for its own components, such as compiler, script monitoring, runtime software or connection management.

While PLC code usually has a durability of 10 years plus and is nearly independent from the development suite version and the OS, this is not necessarily the case for higher level programming, including LabVIEW. Therefore, we limit the changeover of releases to what is absolutely necessary. Further, software projects are managed using professional versioning tools running on

central IT resources, in combination with routine backup copies.

To give some numbers, table 2 shows renewal and upgrade rate we have experienced or can expect in all conscience:

Table 2: Renewal and Upgrade Rates

PLC hardware	> 10 yrs *
PLC application software	~ 10 yrs *
PLC development software	~ 3 yrs
WinCC server and client PCs	~ 5 yrs
WinCC application software	> 5 yrs **
WinCC development software	~ 3 yrs
NI hardware (PC)	~ 5 yrs
NI hardware (PXI)	> 5 yrs ***
NI application software	~ 2 yrs
NI development software	~ 3 yrs

* few PLC systems have been completely replaced by now, they run between 3 and 10 years

** driven by new software options, WinCC applications were reworked in parts

*** PXI system are involved for about 5 years, no complete replacements so far

OUTLOOK: FUTURE CENTRE FOR HIGH INTENSITY BEAMS

From October 2011 on, parts of the ELBE beam lines and user facilities are being dismantled to give room for new installations addressing the scientific needs of our users or improvement of their experimental sites. Including a building extension, we will install a new THz source and laboratory, the revised neutron production target and the revised interaction chamber for high brightness laser beams with ELBE electrons.

For the ELBE CS, this is an ideal opportunity to keep track with recent and already proven developments:

- The aged out S5 system of the PSS is currently replaced by S7 technology. This system will be expanded to the ne building, including interfaces to personnel access management.
- New beam lines will be equipped with newer PLCs, including extensive re-engineering of the PLC software towards better object orientation and efficiency.
- Profibus Technology will be partly replaced by ProfiNET (an industrial Ethernet standard) to provide more convenient data connections between PLCs and the PC/PXI world based on TCP/IP, as well as to improve the coupling performance of the WinCC system.
- As an in-house pilot project, an optical beam line will equipped with EPICS control. Interfaces to WinCC are realized with TCP/IP communication via PLCs.
- The machine protection system fast chain will be completely redesigned in-house, applying pure hardware logics with PLC interface. The overall reaction time will be decreased below 1 ms, according to the demands of the beam power upgrade (64 kW maximum). PLC interrupt handling is limited to about 1.5 ms and will not be involved in this system anymore

SUMMARY

When looking at implementation time and maintenance, the ELBE control system with its mainly commercial components and in-hose developed software is a good choice for a facility of this size with around 25 staff members running in full 24/7 user operation. The price of this is a large variety of interfaces and system constraints to be managed, and the lack of tools specifically designed for accelerator physics. The aimed future role of ELBE as a facility with > 50 % external users demands more system openness and data availability than we can provide by now, and will more and more involve industrial and office IT.

REFERENCES

[1] IEC 62242-1 “Enterprise control system intzegration – Part 1: Models and Terminology”, Geneva, Switzerland, 2003 (http://www.iec.ch)
[2] http://www.automation.siemens.com
[3] http://www.beckhoff.com/
[4] http://www.ni.com
[5] http://www.opcfoundation.org/

THE PERFORMANCE TEST OF F3RP61 AND ITS APPLICATIONS IN CSNS EXPERIMENTAL CONTROL SYSTEM*

Jian Zhuang[#1,2,3], Kejun Zhu[1,3], Yuanping Chu[1,3], Jiajie Li[1], Lei Hu[1] , Dapeng Jin[1,3]
Institute of High Energy Physics[1]
Graduate University of Chinese Academy of Sciences[2]
State Key Laboratory of Particle Detection and Electronics[3]

Abstract

F3RP61 is an embedded PLC CPU module developed by Yokogawa, Japan. It is based on PowerPC 8347 platform. Linux and EPICS can run on it. We do some tests on this device, including CPU performance, network performance, CA access time and scan time stability of EPICS. We also compare F3RP61 with MVME5100, which is most used IOC in BEPCII and BESIII. After this tests and comparison, the performance and capability and role of F3RP61 in CSNS (China Spallation Neutron Source) experimental control system is clear. It can be used as communication nodes between device control layer and global layer. And in some cases, F3RP61 also has the capability to exert more functions such as control tasks.

INTRODUCTION TO TARGET AND INSTRUMENTS OF CSNS

Neutron scattering is a powerful method to probe the structure of the microscopic world, becoming a complementary technique to x-ray in the advanced researches in physics, chemistry, biology, life science, material science, new energy, as well as in other applications. To meet the increasing demands from user community, China decided to build a world-class spallation neutron source, called CSNS. It can provide users a neutron scattering platform with high flux, wide wavelength range and high efficiency. The pulsed-beam feature allows studies not only on the static structure but also the dynamic mechanisms of the microscopic world.

CSNS mainly consists of an H- linac and a proton rapid cycling synchrotron. It is designed to accelerate proton beam pulses to 1.6 GeV kinetic energy at 25 Hz repetition rate. Proton pulses strike a solid metal target to produce spallation neutrons.

The high-energy proton beam extracted from the accelerator bombards on the target to generate neutrons through spallation reaction.

INTRODUCTION TO CONTROL SYSTEM OF TARGET AND INSTRUMENTS

The control system of Target and Instruments provides global control and monitor to all sub-system in Target and Instruments division.

The control system is divided into 3 layers, including front control layer, local control layer and global control layer. Global control layer and local control layer are based on EPICS software. In CSNS Experimental Control System, YOKOGAWA PLC will be adopted as controller in device control layer. EPICS is a SCADA software widely used in large scale physical experiment and accelerator.

Traditionally, device controller in front control layer is a ladder PLC. Control logic is running in this ladder PLC. A soft IOC based on industrial PC is used to connect global control layer based on EPICS and ladder PLC in front control layer. This soft IOC exchanges information only. Because of using two cascaded controller, this system structure decreases the system reliability and costs a lot. A nature idea that these two controllers are combined into one is coming out.

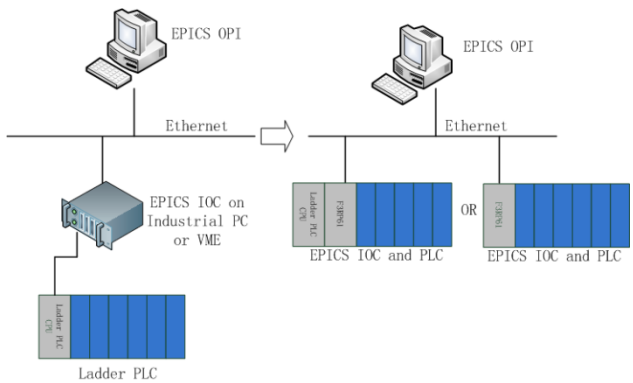


Figure 1: Simplify the path from front control to global control. After Simplifying, reliability increase while cost much fewer.

Now, a new PLC (Programmable Logic Controller) CPU module based on linux OS from YOKOGAWA, F3RP61 [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11], began to be applied in larger physics plant. F3RP61 can be used as EPICS IOC with traditionally ladder PLC CPU in the same framework or even replace the ladder CPU. In way of F3RP61 running in same framework of ladder PLC CPU, the cost of switch and linking cable is saved. In case of F3RP61 replacing ladder CPU, the control logic is running in F3RP61. It can use almost all F3 I/O module directly.This saves the cost of ladder PLC CPU further.

F3RP61 will be used as information exchange and control node in CSNS Experimental Control System. The performance of F3RP61 is the important to design the whole control system.

*Work supported by Institute of High Energy of Physics
Corresponding author: jindp@mail.ihep.ac.cn

INTRODUCTION TO RP61 PLC MODULE AND TEST ENVIRONMENT

F3RP61

F3RP61 [14][15] from YOKOGAWA is based on POWERPC MPC8347 533MHz processor. MPC8347 is an E300 core Soc from Freescale Inc. MPC8347 integrates DDR controller, two 100BaseT Ethernet, USB, serial, and mini PCI interface.

Test Environment

All the tests are done on Yokogawa F3RP61-21. The os is linux-2.6.24.3; c compiler is gcc-3.4.3, and libc is libc-2.3.4. RootFs is on Scandisk 2G CF card in the F3RP61. A Dell PC running scientific linux CERN 4.7 is connect to F3RP61 with Cisco 3750 switch. This Dell PC is used as consol host and communication pair of F3RP61.

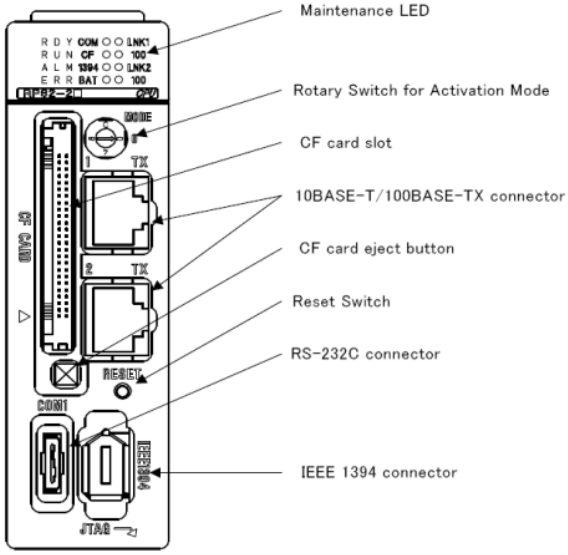


Figure 2: The Front-View of F3RP61.

THE CPU PERFORMANCE TEST

Nbench [12] benchmark is used in performance test to known RP61 CPU well. Nbench is a CPU and memory benchmark program set to test single core CPU, including NUMERIC SORT，STRING SORT，BITFIELD，FP EMULATION，FOURIER，ASSIGNMENT，IDEA，HUFFMAN，LU DECOMPOSITON test program. MEM, INT and FP Performance Index will be obtained by comparing to AMD K6-233 performance after Nbench running. MEM index is for processor bus, cache efficiency and memory performance. INT is integer computing capability of processor. FP is double float performance of CPU. All these expose the theoretical upper limit of the CPU.

The results of Nbench on RP61 are showed in figure 3. These results show that, comparing to AMD K6/233, computing intensive kind of program gain more accelerating.

Figure 4 shows the performance between RP61 and MVME5100, a widely used single border computer in physics experiment and accelerator. Nbench test shows, for CPU capability, performance of F3RP61 is closed to the MVME5100. This means that F3RP61 in compute-intensive applications have certain advantages. Comparing to traditional ladder CPU, RP61, with good operational performance, can do some more complex computing, such as advantage PID algorithm.

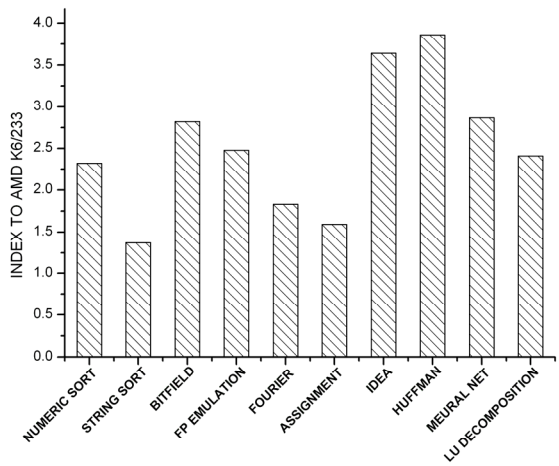


Figure 3: Ten reference program performance Index of Nbench Program Set.

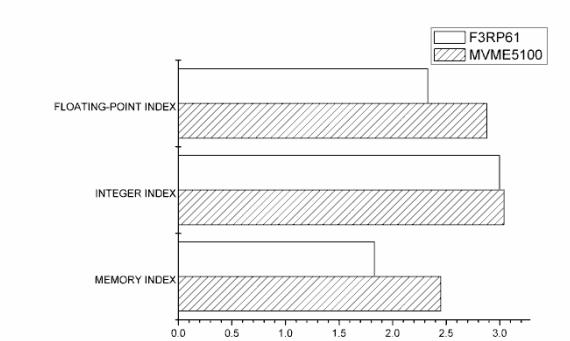


Figure 4: Performance Index Comparison. F3RP61 vs MVME5100.

NET PERFORMANCE TEST

In CSNS Experimental Control System, an important role of RP61 is information exchange node between device control layer and front control layer. So the net throughout performance and CPU utilization is an important index of performance.

Netperf [13] is a standard tool based on TCP and UDP protocol to evaluate net performance. In different test case, Netperf can do bulk data transfer or request/response test. In bulk data transfer test mode, the

result reflects how fast a system can send or receive data to/from another system.

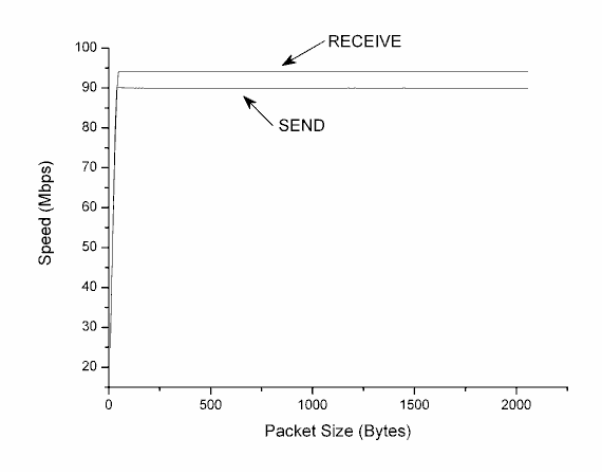


Figure 5: The net speed of F3RP61.

Figure 5 shows the send and receive speed vs. packet size of RP61 to a DELL PC. From about 100 byte, the send and receive speed can close to 100M line speed.

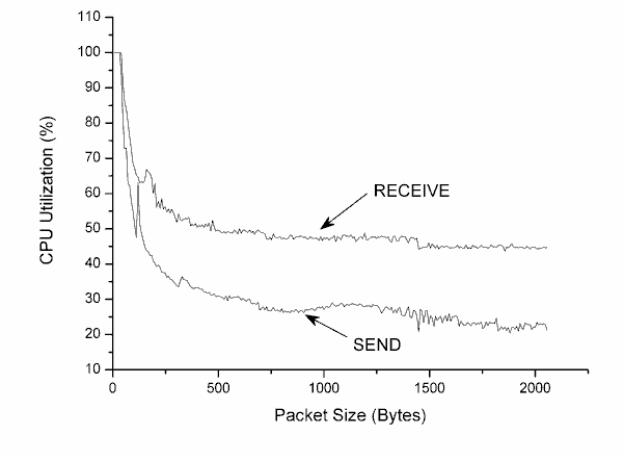


Figure 6: CPU Utilization of sending and receiving.

Figure 6 illustrate the CPU occupation with packet size. In this figure, the speed of net increases with the packet size increasing, while the CPU occupation decrease. This effect is caused by the different packing and depacking cost in TCP/IP stack in different packet size. In control system, the small size packet is dominated, so the net send task should acquire more CPU time. And also from figure 6, as more interruption happens, receive task costs more CPU time than send task. NAPI feature in the linux kernel will reduce CPU utilization obviously.

This CPU utilization of F3RP61 is much larger than MVME5100. This means, the net capability of F3RP61 is not so good.

THE PERFORMANCE ON EPICS

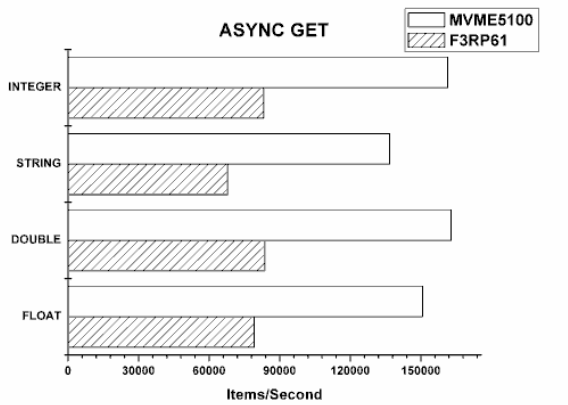


Figure 7: Asynchronous read performance comparison, MVME5100 vs. F3RP61.

In the target and instruments control system of CSNS, F3RP61 is used as information exchange node between device control layer and global control layer, providing PV service and control information exchanging. Besides this, F3RP61 can provide some control functions within epics framework. So how to evaluate and balance these two functions is the problem in front of target and instruments control system.

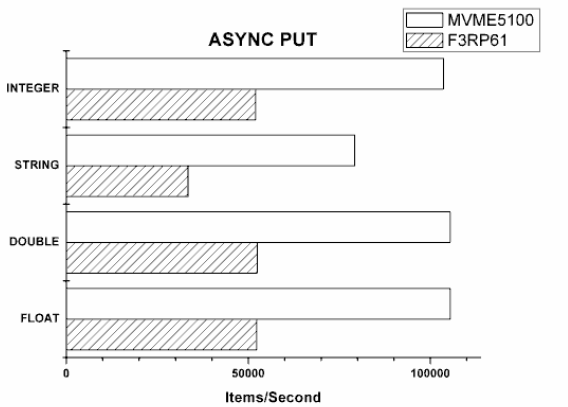


Figure 8: Asynchronous Write performance comparison, MVME5100 vs. F3RP61.

The main tasks of EPICS are providing periodic scan for control and PV access from Ethernet. We can estimate the maximum PV access the F3RP61 can provide through the test to provide design roof for the system design.

EPICS IOC performance on F3RP61 can be estimated through PV access speed test. Figure 7, 8, 9 shows F3RP61 PV access speed comparing to MVME5100. In this test, F3RP61 and Dell PC are connected in the same CISCO 3750 switch. F3RP61 can provide about one half PV access of MVME5100 in case of full CPU occupation. Considering real-time information exchange and reliable PV access service, 20% CPU time can be used to PV

service on information exchange dominant node. More less CPU time, about 5% to 10%, can be allocate to PV access in hybrid node of information exchange and control.

Synchronous read speed is much slower than asynchronous access speed. The reason much time is cost in waiting the response on net.

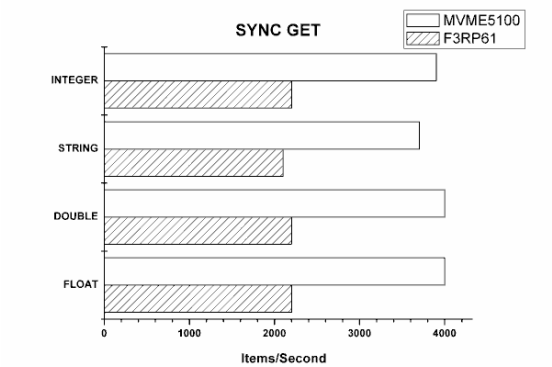


Figure 9: Synchronous read performance comparison, MVME5100 vs. F3RP61.

For control task, the stability of scan period is most important. Figure 10 depicts the output of IOC period scan on F3RP61. The scan period in this chart is 5ms. The period of output is 10ms. The default EPICS configuration was used. After more than 190,000 samples, the Std. Dev. of scan period is about 26us. The Max-min jitter is about 600us. For the general control task, the typical period is about tens of millisecond. This deviation can be accepted.

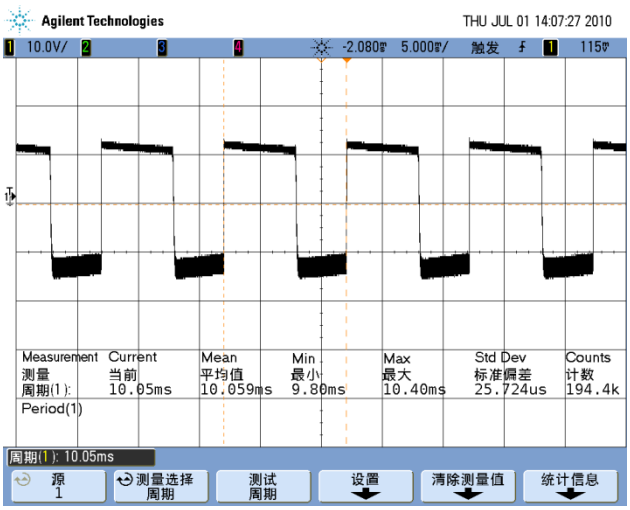


Figure 10: Jitter of 10ms period.

CONCLUSION

Through all the tests in this paper, the conclusion that F3RP61 can be used as information exchange and control node in the target and instruments control system of CSNS can be drawn. Comparing to traditional ladder CPU, F3RP61 is able to do more computing，to run EPICS and to provide more information exchanging.

REFERENCES

[1] A. Uchiyama et al. “Development of Embedded EPICS on F3RP61-2L”, Proc of PCaPAC2008, Ljubljana, Slovenia, Oct. 2008, pp.145-147

[2] S. Araki et. al., “Magnet Power Supply Control Based on EPICS on F3RP61 for KEK-ATF”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009,

[3] S. Motohashi et. al., “Data Acquisition System of Beam Loss Monitors of J-PARC Main Ring”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009,

[4] M. Komiyama et al., “Upgrading the Control System of RIKEN RI Beam Factory for New Injector”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009, pp.277-279

[5] M. Takagi et al., “Control of the J-PARC Slow Extraction Line Based on Embedded EPICS”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009, pp.549-551

[6] T. Nakamura et al., “Upgrading the Control System of the Movable Masks for KEKB”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009, pp.546-548,

[7] K. Mikawa et al., “Embedded EPICS Controller for KEKB Pulsed Quadrupole Magnet Power Supply”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009, pp.558-560,

[8] J. Odagiri et al., “Application of EPICS on F3RP61 to Accelerator Control”, Proc of ICALEPCS2009, Kobe, Japan, Oct. 2009, pp.916-918,

[9] C.Y. Wu et al., “Control of the Pulse Magnet Power Supply by EPICS IOC Embedded PLC”, Proc of IPAC2010, Kyoto, Japan, May. 2010, pp.2731-2733

[10] T. Kosuge and K. Nigorikawa, “STARS on PLC, Proc of PCaPAC2010”, Saskatoon, Canada, Oct. 2010 pp.73-75.

[11] J. Odagiri et al., “Development of Image Processing System on Embedded EPICS for Beam Diagnostics”, Proc of PCaPAC2010, Saskatoon, Canada, Oct. 2010, pp.165-167

[12] http://www.tux.org/~mayer/linux/bmark.html

[13] http://www.netperf.org/netperf/

[14] RTOS-CPU module (F3RP61) Linux BSP Reference Manual, Yokogawa, Japan

[15] RTOS-CPU Module (F3RP61) Hardware Instruction Manual, Yokogawa, Japan

AGILE DEVELOPMENT AND DEPENDENCY MANAGEMENT FOR INDUSTRIAL CONTROL SYSTEMS

B. Copy, M. Mettälä, CERN, Geneva, Switzerland

Abstract

The production and exploitation of industrial control systems differ substantially from traditional information systems; this is in part due to constraints on the availability and change life-cycle of production systems, as well as their reliance on proprietary protocols and software packages with little support for open development standards [1]. The application of agile software development methods therefore represents a challenge which requires the adoption of existing change and build management tools and approaches that can help bridging the gap and reap the benefits of managed development when dealing with industrial control systems. This paper will consider how agile development tools such as Apache Maven for build management, Hudson for continuous integration or Sonatype Nexus for the operation of "definite media libraries" were leveraged to manage the development life-cyle of the CERN UAB framework [2], as well as other crucial building blocks of the CERN accelerator infrastructure, such as the CERN Common Middleware or the FESA project.

INTRODUCTION

Agile software development methodology characterizes any development method that emphasizes incremental deliveries, working software and fast response to change. It emerged in the 1990s as a response to so-called heavyweight methods, relying on long running project plans, waterfall models and micromanagement.

Agile methods such as SCRUM or Extreme Programming rely on the old maxim “a problem shared is a problem halved”, by encouraging the collaboration of small but cohesive development teams performing fast evolving deliveries. Teams collaborating on a large project will therefore continuously contribute project deliverables to a “common pot”, from which a best-of-breed product release can be composed at any time.

While agile methods emphasize the importance of human interactions over automated processes and customer collaboration over contract negotiations, it is not surprising that software tooling quickly emerged to support operating in such dynamic and fast-paced environments.

Agile Tooling

It would be misleading to consider that agile development can be effective solely with good will, highly motivated independent teams and supporting customers and stakeholders. Agile software tooling aims at freeing software developers from repetitive tasks, while providing immediate and up-to-the-minute feedback on the status of their team's efforts. Developers can therefore confidently test software changes, share their latest progress with their fellow team members without any manual intervention and in the end focus on providing fast updates and concentrating on fulfilling customer requirements.

Agile tooling is necessarily based on software development processes that in most part, originated with heavyweight project management approaches.

SOFTWARE DEVELOPMENT PROCESSES

In order for agile teams to collaborate in a streamlined and manageable manner, well-known project management processes such as :

- Issue management
- Change management
- Dependency management
- Release management

must constantly collaborate and exchange information.

A short summary of each will help us understand their importance, and provide examples of software packages used at CERN to fulfil such demands.

Issue Management

This process focuses on collecting, prioritizing and refining customer demands and internal product quality feedback. Whether use cases, requests for new features or defects identified in existing products, such inputs must be classified and scheduled so as to reduce the risk of unwanted side effects and ensure timely release delivery. Change brought to a product must always take for reference a prior request for such change (once again, to reduce the risk of unidentified and unwanted modifications that could break a working product).

Change Management

This process ensures that changes can be audited, and grouped into coherent set of modifications. Sets of modifications will eventually compose a release. Change management is often seen as a burden by software developers, but becomes a key activity when it is coupled with release management – for instance, once a product version is out in the wild, it is essential to know exactly what differentiates this version from the one that worked better a few weeks or months ago.

Dependency Management

As agile teams happily release ever improving deliverables, being able to orchestrate dependencies among these deliverables becomes an increasingly difficult task. Establishing clear policies and allowing the definition of dependency ranges (*e.g.* by specifying that

Project A can tolerate any of the latest version of Project B, starting from version 1.5 up to version 2.0 and anything in between) becomes a very important aspect of software project management. Considering that software project increasingly rely on third-party open source dependencies, it also becomes important to state and monitor the provenance of such dependencies (*i.e.* if two versions of the same library are available on the internet, one from an untrusted source and one from the official provider of the library, it is naturally preferable to obtain it from the official provider).

Release Management

In order to facilitate software reuse (one of the cornerstones of agile development), software artifacts must be made available in a readily consumable form, so that no time needs to be spent rebuilding such artifacts using the original source (which, if change and dependency management are kept consistent through an established release policy, should of course always be possible). In ITIL v3 [3], a library of best practices in software infrastructure management, a repository used to store readily consumable artifacts is referred to as a **Definite Media Library** (DML). A DML acts as a reference for all artifacts maintained by an organization, and supports features such a search indexes, structured navigation and queries, metadata management and access rights policies enforcement.

Release management also becomes relevant in order to deliver automatic updates to software end-users. Auto-update capabilities have become a common feature of modern operating systems and software packages aimed at the general public.

As an example, the UAB toolset [5] (a development suite for the generation of industrial control systems) offers a Bootstrap utility, illustrated in Fig. 1, which upon startup verifies whether any new versions of the UAB components (such as new device type libraries, new generation plugins etc...) have become available in a central repository.

If new UAB components have been released, end users are offered the possibility to download these components and use them for the generation of their application.

Sonatype Nexus, a commercial DML implementation that acts as an Eclipse P2 repository and a Maven repository, offers a programmatic API (named Aether) which allows to query and download repository contents. The UAB Bootstrap component for instance relies on the Aether API to determine whether new UAB components (in effect, Maven artifacts deployed in Nexus) are available and to download them.

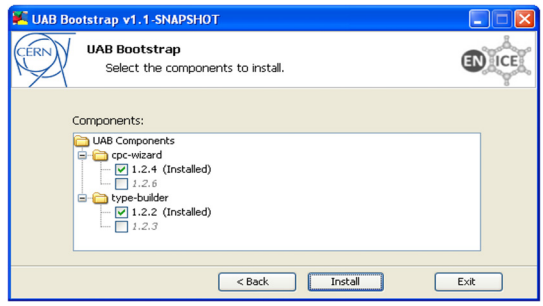


Figure 1: The UAB Bootstrap (using the Aether API).

Continuous Integration

Continuous integration is a quality process that provides constant feedback on changes performed by developers. Coupled with change management, it allows at any point to ensure that developers are informed of regressions (or build breakage) they might introduce in their project.

PROCESSES INTEGRATION

In the context of the UAB project and the LHC Accelerator complex, CERN employs a variety of software packages that support the previously enumerated processes, including :

- **Atlassian JIRA** for issue management
- **Subversion** (SVN) for change management
- **Apache Maven** build for dependency management
- **Sonatype Nexus** as a *definite media library* and release repository for Maven and Eclipse dependencies.
- **Hudson / Atlassian Bamboo** for continuous integration

This best-of-breed selection mixing open-source software with commercial tools operating under various licenses poses necessarily some integration issues. Navigating from one information source is obviously quite possible, as each tool provides navigation links of some sort to its peer processes – but happens to be quite cumbersome when investigating regression issues, plain bugs or releases of insufficient quality due to failure in change or issue management processes.

The CSAR project therefore defines a data format and offers tools that greatly simplify data extraction, navigation and visualization of engineering process related information (when investigating the cause for the introduction of a regression or the presence of a new bug).

Resource Description Framework (RDF)

RDF is a meta-model introduced by the W3C to unify data and meta-data in web-based documents. RDF relies on subject-object-predicate expressions to embed meta-meaning in ordinary data, allowing the data to be processed and leverage by a machine.

RDF acts as an ideal *lingua franca* between our various engineering process support tools, allowing us to harvest all our information and collate it in a navigable way.

Data Visualizations and the Exhibit Framework

Data visualizations represent large amounts of information in an immediately understandable form. Maps, pie charts or tables are traditional visualizations, but others such as heat maps or sparklines can also deliver great expressiveness. In any case, visualizations are decoupled from the data they represent.

The MIT Exhibit framework offer a generic way to represent, query, filter and visualize RDF data. The CSAR project leverages Exhibit and integrates a new visualization yet not supported by the framework for the representation of dependency trees.

Exhibit Reports Build Integration

The CSAR project provides an Apache Maven site plugin – which seamlessly integrates engineering process information into a web-based Maven documentation site [4].

This is particularly useful for continuous integration software packages that support Maven but would not necessarily understand RDF or be able to handle the Exhibit framework.

Fig. 2 presents an example of a Maven generated documentation site integrating an Exhibit report. Once RDF data is provided to Exhibit, the framework automatically generates the required user interfaces elements such as filters and navigation links.

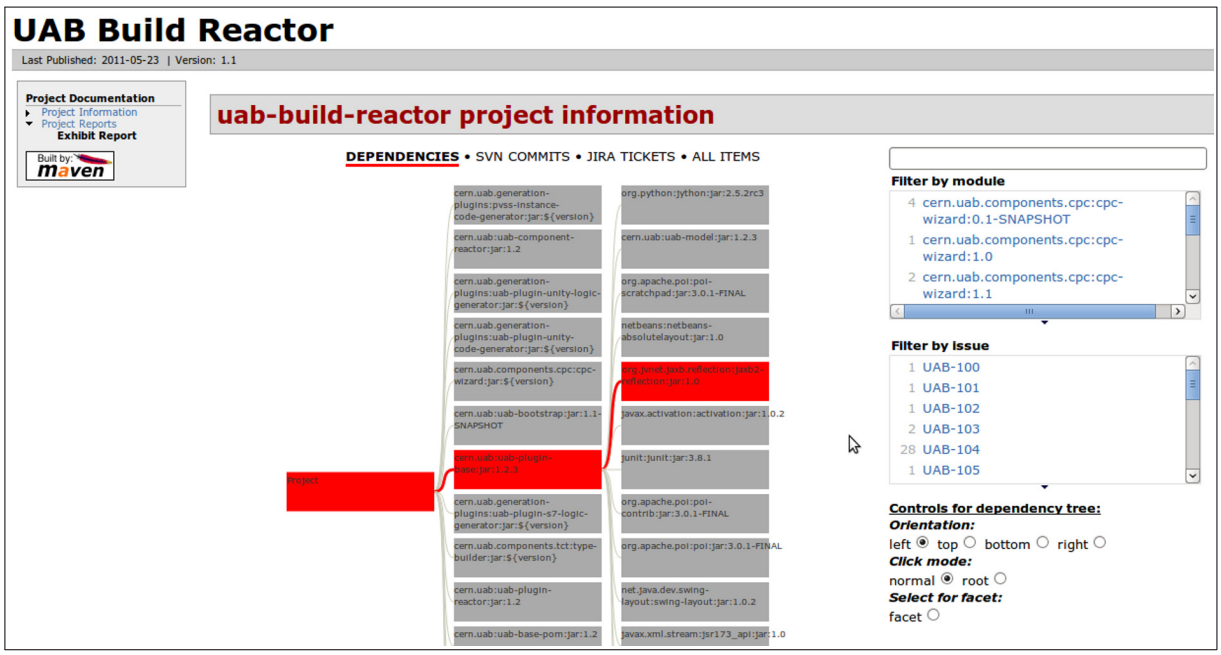


Figure 2: Web-based Maven-generated documentation site with interactive dependency tree and filter fields.

The Exhibit framework also introduces new visualization concepts such as the Timeline visualization, a Javascript widget that allows to browse time-based events, making it particularly suitable for reviewing SVN commit activity or JIRA activity.

Fig. 3 and 4 provide examples of Timeline visualizations as generated by Exhibit according to the RDF data collected by the CSAR collection plugins.

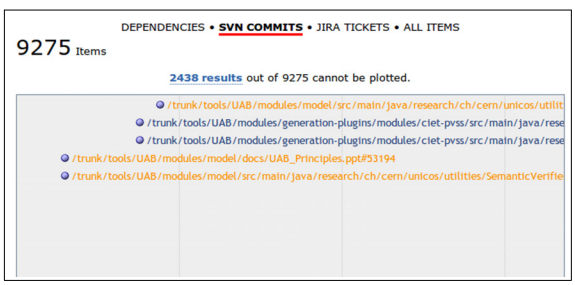


Figure 3: Timeline visualization of SVN activity.

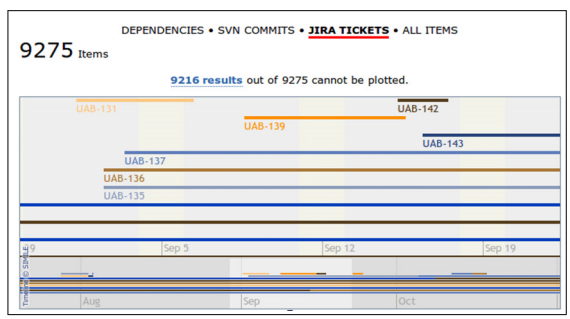


Figure 4: Timeline visualization of JIRA activity.

Timeline visualizations in the Exhibit framework provide navigation links and popup windows which are automatically derived from RDF data.

DATA INTEGRATION

The CSAR project provides Java executables that can be integrated as part of a build process (using their respective Maven plugin wrappers) and extract RDF

information from our various engineering process support systems (JIRA, Maven metadata, Nexus, Subversion).

Once in a unified RDF format, data integration is a matter of collating files together and transforming them to web-friendly RDF dialects (known as Exhibit JSON).

The generation of Exhibit user interface and navigation links is then simply a matter of configuration.

CONCLUSIONS

Integrating software engineering process data sources is useful in order to obtain a global vision of development and release activities. While most engineering process deliver access to structured data, integrating and correlating this information is currently a non-trivial task.

While our usage of RDF as a common data format has certainly proven a workable approach, certain limitations subsist in the current support of RDF. The Exhibit framework for instance was designed highly interactve representation of low volumes of information. A sample extraction of the UAB project activity over the course of 14 months, totalling about 600K lines of code, yielded a 5 megabytes Exhibit JSON input file containing 9275 records, which the Exhibit framework handles with difficulty (requiring a 20 seconds initial startup time and delivering occasional sluggish data filtering performances).

A complete rewrite of the Exhibit framework (Exhibit v3) is interestingly under way in order to ease the integration of third party visualizations and make it able to cope with datasets gathering more than 5 millions of records.

REFERENCES

[1] R. Barillère, Ph. Gayet, *"UNICOS A Framework to build industry-like control systems, principles and methodology ",* ICALEPCS 2005, Geneva, Switzerland, WE2.2-6I

[2] M. Dutour, "Software factory techniques applied to Process Control at CERN", ICALEPCS 2007, Knoxville Tennessee, USA, http://www.JACoW.org.

[3] APM Group, "*What is ITIL ?*", 2007-2011, http://www.itil-officialsite.com/AboutITIL/WhatisITIL.aspx

[4] Apache Maven Project, "*Guide to creating a documentation site*", http://maven.apache.org/guides/mini/guide-site.html

[5] B. Copy et al., "Model Oriented Application Generation for Industrial Control Systems", ICALEPCS 2011, Grenoble, France, WEAAULT02

[6] Massachusetts Institute of Technology (MIT), "*The Exhibit Framework*", 2011, http://www.simile-widgets.org/exhibit3/

QUICK EXAFS EXPERIMENTS USING A NEW GDA ECLIPSE RCP GUI WITH EPICS HARDWARE CONTROL

R. J. Woolliscroft, C. Coles, M. Gerring, M. Pearson, Diamond Light Source, Oxfordshire, UK

Abstract

The Generic Data Acquisition (GDA) framework [1] is an open source, Java and Eclipse RCP [2] based, data acquisition software framework for synchrotron and neutron facilities. A new implementation of the GDA on the B18 X-ray beamline at the Diamond synchrotron will be discussed. This beamline performs energy scanning X-ray Absorption Spectroscopy (XAS) experiments. It includes a continuous-scan mode of the monochromator synchronised with various detectors for Quick EXAFS (QEXAFS) experiments. XAS energy scans may now be performed in 30s, where the equivalent step scans take approximately 40 minutes.

A new generic perspective for the GDA client has been developed in which graphical editors are used to write XML files which hold experimental parameters. The XML files are marshalled by the GDA server to create Java beans used by Jython scripts which run on the GDA server. Underlying this, a new generic continuous scanning mechanism for the GDA has been developed. The new Eclipse RCP UI, the new continuous scanning mechanism, new hardware development and integration between the two systems shall be covered.

INTRODUCTION

XAS is a mature X-ray technique, with a wide range of applications across scientific disciplines. As a result, there is a need at modern synchrotrons to provide robust and highly automated XAS beamlines, designed to be used by both experienced and occasional users.

B18 is a general purpose XAS photon beamline, complimentary to other XAS beamlines at Diamond. A new quick scanning mode has been developed in which EXAFS data can be collected with a continuous move of the monochromator.

The experimental hutch has two sample positions; an in-vacuum position for 2-4keV (soft X-ray) and an in air position for 4-36keV (hard X-ray). The hard X-ray position has a large experimental area for a range of sample environments (Fig. 1).

DETECTOR

Fluorescence data for EXAFS studies is collected using a 9 element XSPRESS Ge detector [3]. This detector can collect at high count rates whilst being placed close to the sample to maximise the fluorescence yield.

The detector is controlled by a Time Frame Generator (TFG2) [3] card which has a number of programmable input and output signals. The TFG2 card is used to receive and fan-out signals from the monochromator Bragg motor during a quick scanning data collection (Quick EXAFS or QEXAFS). It can be used to drive other detectors which may be used during the experiment. For the QEXAFS experiments, most commonly used are the ion chambers which provide a normalisation for the fluorescence yields.

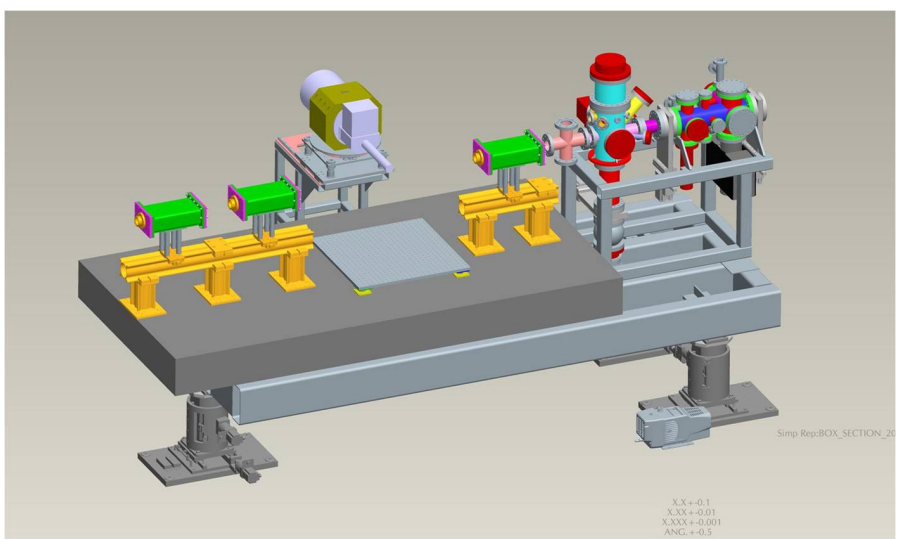


Figure 1: Diagram of the B18 experimental hutch. The in-air sample location is shown. This has a large area for a range of sample environments. The XSPRESS detector for fluorescence and the ion chambers for transmission measurements are shown.

HARDWARE FOR QEXAFS EXPERIMENTS

The monochromator Bragg axis is controlled using a DC brushless in-vacuum motor, with four encoders placed at equal distance around the stage. There is also an encoder on the back of the motor. A Delta Tau Geobrick Low Voltage Integrated Motion System (LV IMS) is used as the motor controller, and this is the Diamond standard controller for beamlines currently being constructed. The Bragg axis and crystal distance axis are controlled together to provide an 'energy' combined axis, and these calculations are performed directly on the controller. This provides us with true synchronised motion of Bragg and crystal distance. Using this control method a very smooth continues motion between two energy points can be achieved, at speeds of up to 1deg/s.

The Experimental Physics and Industrial Control System (EPICS) [4] is used to control and set up the motor controller. Each axis controlled by the Geobrick has an EPICS Motor Record interface, which provides a consistent 'motor' interface for the GDA. The same interface is used for the virtual 'energy' axis, which means the GDA can treat the energy axis as any other axis, and scan it in the same way.

The EPICS Input Output Controller (IOC) is also responsible for setting up the 'position compare' trigger signals for the Bragg axis. These are TTL pulses produced by the Geobrick as the Bragg axis encoders increment in position. The triggers are fed directly into the TFG2 card, which is controlled by the GDA. The readout of the ion chambers is performed using a GE-Fanuc 110BL VME ADC card. This is a 24-bit sigma-delta ADC running at up to 108KHz, which is read out through a new driver used by an IOC running VxWorks.

The trigger signals that are fed into the TFG2 card are used to provide a 'gate' signal. This signal is read into one of the ADC channels, and is used to provide a means by which the IOC can calculate mean and standard deviation of the samples collected during the gate period. Gates as long as several seconds and as short as 1ms are used. Buffering of this data is provided by the EPICS IOC in arrays, so that they can be read out asynchronously by the GDA. However, if required, raw ADC count data is also provided by the EPICS IOC.

NEW CONTINUOUS SCAN

The GDA has a flexible and powerful step scan mechanism which has been applied to the full range of techniques used at the Diamond synchrotron. A new generic continuous scan mechanism was developed for implementation on the B18 beamline, but which could easily be applied to other beamlines with suitable hardware.

Existing Scanning Mechanism

The generic scanning mechanism is available through the *scan* command in the GDA Jython environment. Behind this command is a single class (*ConcurrentScan*) which operates both hardware and detectors in n-dimensional step scans. This operates the hardware through two high-level interfaces: *Scannable* and *Detector*.

The Scannable interface enables the scanning mechanism to operate those objects to be incremented during the scan, and to know what information is supplied by that object at each point. Scannables can represent a wide variety of objects; simple motors, compound motions or even offline data processes which return the results of their calculations asynchronously.

The Detector interface is an extension to the Scannable interface. At each point in the scan Detector objects are used to collect data after the Scannable objects have completed their step movement.

Working with the ConcurrentScan class is a system to broadcast the data from each step in the scan to parts of the UI for live data display to the user and to file writers to record the data in various data formats. The performance of this system is fast enough to cope with large and frequent data points at the speed expected in B18 QEXAFS scans.

Extension to the Scanning Mechanism

The new continuous scan mechanism is designed to work within the existing step scan mechanism. It incorporates many aspects of that system to maximise code reuse. Its design has analogous interfaces: ContinuouslyScannable and BufferedDetector, which are extensions of the Scannable and Detector interfaces respectively.

There is a new single generic continuous scan class (ContinuousScan) which uses those generic interfaces. This class extends the ConcurrentScan class so the continuous scan may be used as the innermost dimension of an n-dimensional step scan. For example, from a single command, the user may loop over a series of samples in an automated changer and perform a QEXAFS scan on each one.

B18 implementation

An implementation of the ContinuouslyScannable interface was written for B18 which operates the Bragg motor via EPICS and controls the position compare parameters. The XSPRESS detector is operated by an implementation of the BufferedDetector interface from which the amount of data stored and the data itself can be retrieved asynchronously by the scan class.

In the B18 QEXAFS scan, the ContinuousScan class is given the experimental parameters and references to the Bragg ContinuouslyScannable and XSPRESS BufferedDetector objects. The hardware is prepared for the scan, and once the Bragg motor motion has been started the scan class polls the XSPRESS BufferedDetector for new data. As new data becomes available, then the existing step scan's mechanism for recording and visualizing the data is used.

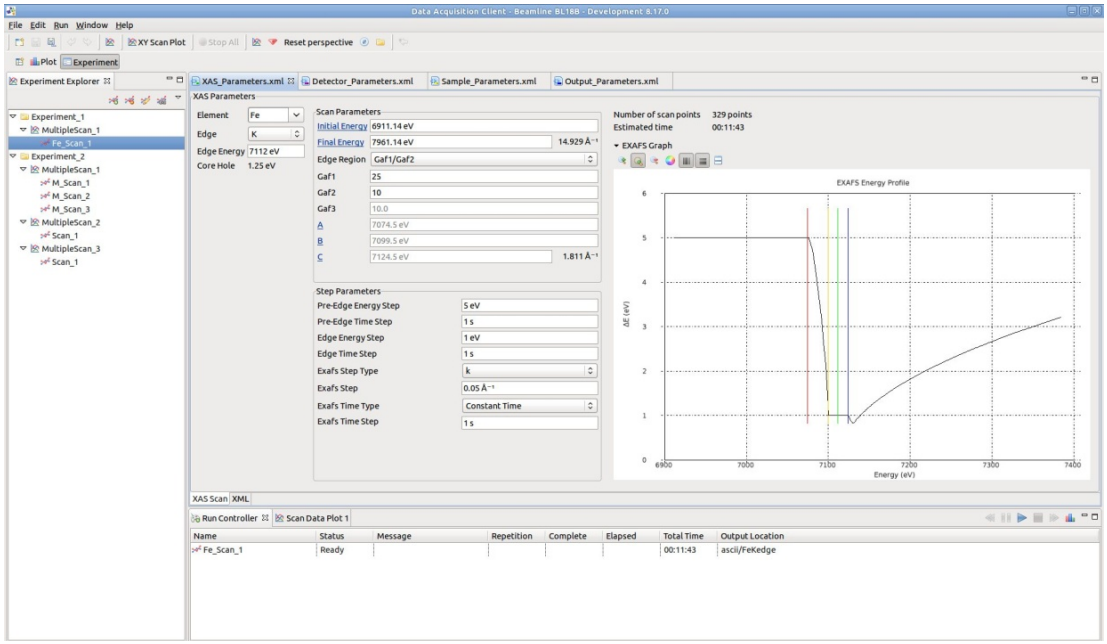


Figure 2: New Experiment perspective for users to design experiments. In B18's implementation four graphical editors are used to write the four XML files which define each experiment. The visible editor is the EXAFS step scan editor.

NEW RCP UI

B18 was the first newly commissioned Diamond beamline to use a GDA client based on the Eclipse RCP platform. A range of generic perspectives have been developed for tasks such as visualisation of data files and Jython scripting.

A new generic perspective, shown in Fig. 2, was written which uses graphical editors to allow users to edit XML files holding the experimental parameters. The perspective has views to manage the XML files and to run a single or a group of experiments (*multiscans*). Multiscans allow several experiments to run sequentially. Multiscans may be stored in different file system folders to keep the files organised. The UI provides a tree view of the multiscans and functionality such as drag and drop to move individual experiments between multiscans. There is a view to run the selected multiscans and display progress.

The perspective is configured using Eclipse extension points and may be applied to other beamlines which could use XML files to define each experiment.

For B18 there are four XML files to describe each experiment. These hold the scan parameters, detector options, sample environment setup and readout options. There are three different editors which write scan parameters to reflect the different experiments run on the beamline: EXAFS, XANES and QEXAFS (detail shown in Fig. 3).

Design

To collect data the UI unmarshalls the XML files into Java beans. These beans are used in a Jython script running on the GDA server which uses the information in those beans to set up the hardware and run appropriate scan commands.

Data is returned to the UI for display and analysis in the same manner whether the scan is a quick or regular EXAFS.

A custom perspective has been written to view experiments in progress, as shown in Fig. 4. This contains views to display the current experiment progress; XSPRESS count rates; visualisation of the data as it becomes available and online data analysis. Graphs show standard XAS analyses: background subtraction, first derivative of the data and the Fourier transform.

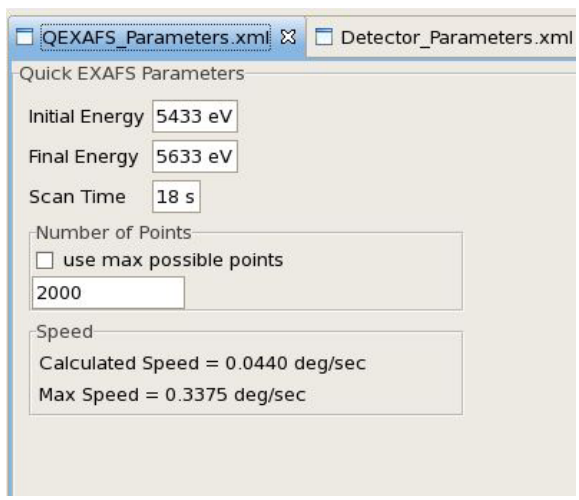


Figure 3: Detail of the Experiment perspective showing the QEXAFS editor.

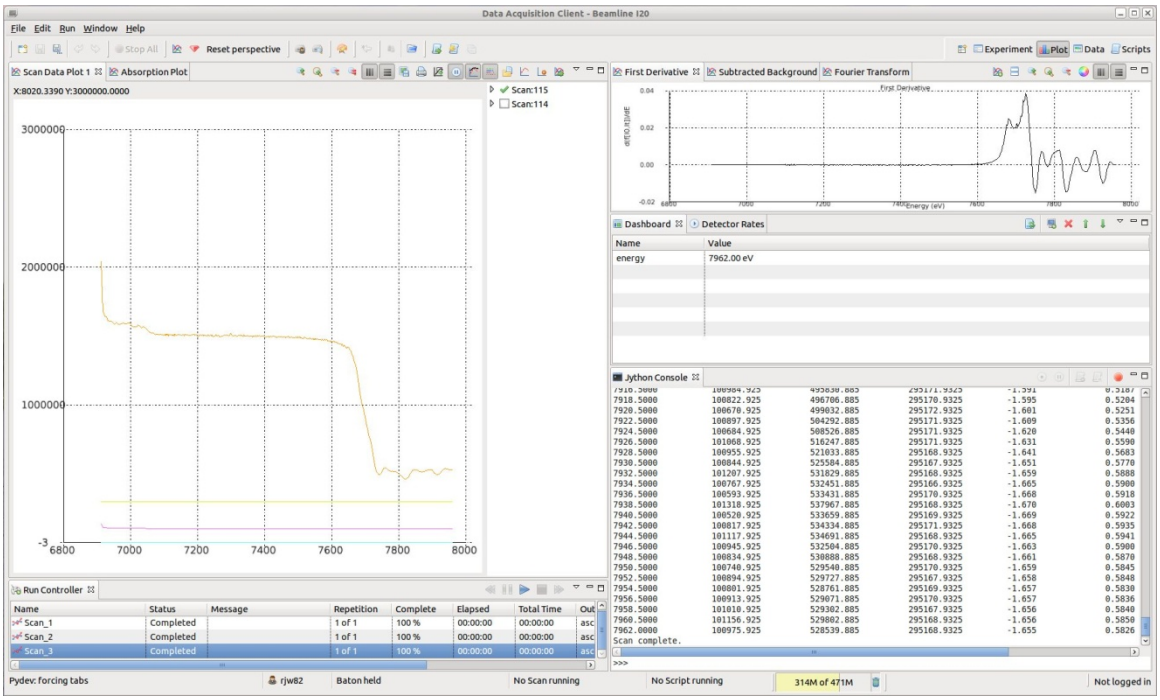


Figure 4: The Plot perspective for Diamond XAS beamlines. This shows data from the experiment in progress. Live plots of the absorption, first derivative of the data after background subtraction, the subtracted background and FFT are shown in other views. The Run Controller view from the Experiment perspective is also shown so experiments may be started from within this perspective.

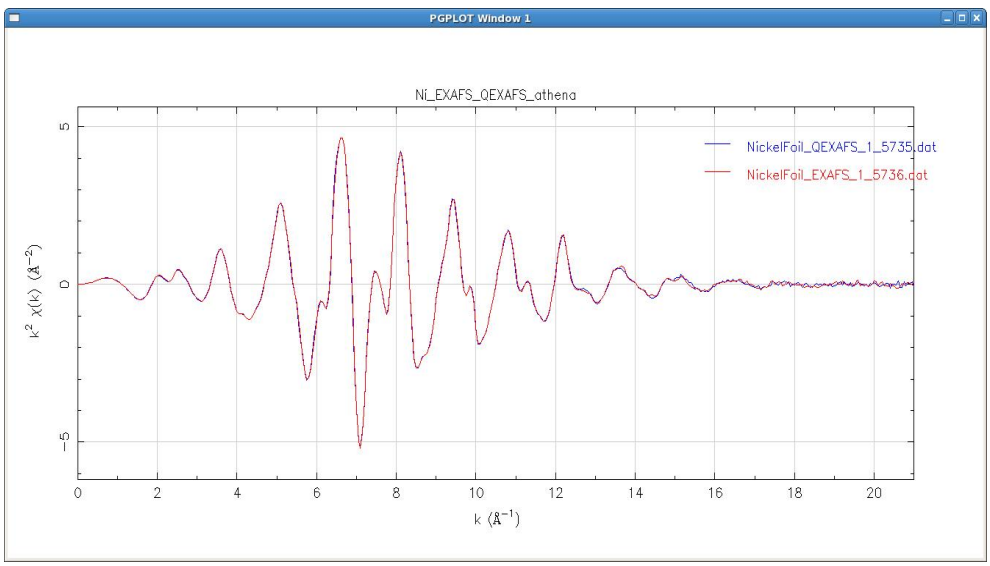


Figure 5: Figure shows the first Quick EXAFS spectrum taken in 60s (25ms per point) (red curve), compared to a step scan running for 50 minutes (1-5s per point) (blue curve).

PERFORMANCE

A typical time for a QEXAFS scan is about 30s where typical step scans take 20 to 40 minutes. The QEXAFS results have been shown to be comparable to step scan data (see Fig. 5).

The QEXAFS mode has been available since Spring 2011 and is currently requested by about a third of proposals to the beamline.

REFERENCES

[1] http://www.opengda.org
[2] http://www.eclipse.org
[3] R. Farrow et al., "XSPRESS: A new generation of detector system for EXAFS studies", Physics B 208 & 209, pp. 256-8 (1995).
[4] http://www.aps.anl.gov/epics/

AN OBJECT ORIENTED FRAMEWORK OF EPICS FOR MICROTCA BASED CONTROL SYSTEM

Z. Geng#, SLAC, Menlo Park, California, U.S.A.

Abstract

EPICS (Experimental Physics and Industrial Control System) is a distributed control system platform which has been widely used for large scientific devices control like particle accelerators and fusion plant. EPICS has introduced object oriented (C++) interfaces to most of the core services. But the major part of EPICS, the run-time database, only provides C interfaces, which is hard to involve the EPICS record concerned data and routines in the object oriented architecture of the software. This paper presents an object oriented framework which contains some abstract classes to encapsulate the EPICS record concerned data and routines in C++ classes so that full OOA (Objected Oriented Analysis) and OOD (Object Oriented Design) methodologies can be used for EPCIS IOC design. We also present a dynamic device management scheme for the hot swap capability of the MicroTCA based control system.

INTRODUCTION

Object oriented analysis and design method has become more and more popular in software engineering field, especially after the appearance of UML (Unified Modelling Language) [1] and MDA (Model Driven Architecture) [2].

EPICS has introduced object oriented (C++) interfaces to most of the core services including the operating system abstraction layer. But the major part of EPICS, the run-time database, only provides C interfaces, which is hard to involve the EPICS record concerned data and routines in the object oriented architecture of the software.

The Object Oriented framework for EPICS (OOEPICS) provides several base classes to encapsulate the details of the record processing. The EPICS records and the devices are designed as objects which enable the direct translation of the UML model into EPICS applications. The framework hides most of the details of the EPICS which enable the user to develop his EPICS device driver without knowing much of EPICS.

The OOEPICS framework was originally designed for a MicroTCA based control system which requires dynamic objects creation and deletion to support the hot swap capability of the system. The EPICS base was extended to be able to load the EPICS records dynamically during run time.

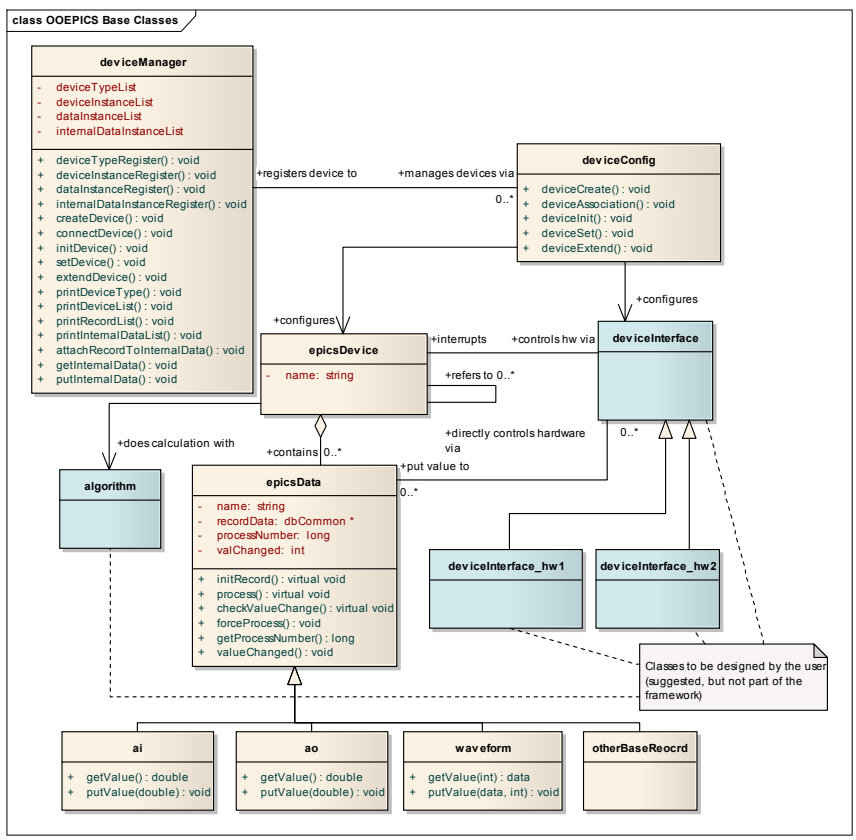


Figure 1: Base classes of OOEPICS framework.

#gengzq@slac.stanford.edu

OOEPICS FRAMEWORK

The OOEPICS framework consists of several base classes and a tool for source code generation which provides a template for the user application.

Base Classes

Figure 1 shows the base classes of the OOEPICS framework.

The base classes setup the basic architecture of the software. Any EPICS applications using the OOEPICS framework can be designed by inheriting the base classes for specific devices.

- epicsData is the base class for EPICS records and device support routines, which realizes some common functionality such as creating EPICS database when the object is created. From that, child classes are derived for all basic EPICS records, such as ao, ai, waveform and so on.
- epicsDevice is the base class for the devices to be controlled which may contains epicsData objects as interfaces to the Channel Access clients.
- deviceConfig realizes common interfaces to configure a device, including registering the device type, creating instances of the device objects and initializing the devices like starting up the threads for the device control.
- deviceManager manages all devices and provides IOC shell commands for device management.

EPICS Records

The key part of the OOEPICS framework is to define an object for each record. The EPICS records were designed more or less based on the object oriented concepts. Table 1 compares the EPICS records and the C++ classes.

Table 1: EPICS Records and C++ Classes

EPICS Records	C++ Classes
Fields	Attributes
Record Support	Methods (platform independent)
Device Support	Methods (platform dependent)

The class of epicsData is defined for the EPICS record. They are linked together by the record name. The object of the class will load the record and obtain the address of the record data during creation and the record will get the address of the object and use its methods as device support.

EPICS Devices

The epicsDevice class is an empty class where the user codes need to be implemented for the control of specific devices.

The epicsDevice class may contain of:

- epicsData objects for uplink control and monitoring.
- Special logic to perform the device control, probably state dependent.
- Algorithms to do signal processing, calibration, optimization and so on.
- Common logical interfaces to interact with physical device.

Device Manager and Configuration

The deviceManager and deviceConfig provide a common way to create, associate, initialize and set up a device control module in user IOC.

The deviceManager class also provides the EPICS API that can be called in the IOC shell. In principle, the user does not need to create any IOC shell commands.

The deviceConfig class should be realized by the developer for specific devices.

Code Generation

The tool of "epics_driver_template" is designed to generate the source code template from the EPICS database file.

The tool will compile the database file into C strings and then use them to initialize the epicsData objects. A derived device class will be generated with the epicsData objects as attributes. A derived deviceConfig class will be generated which allows to define the procedure to create, initialize and setup the device objects.

Development Procedure

Figure 2 shows the procedure for developing EPICS device driver modules with the OOEPICS framework.

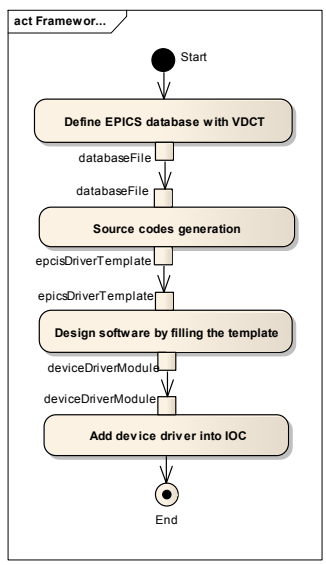


Figure 2: Design procedure with OOEPICS.

DYNAMIC RECORDS LOADING

MicroTCA is hot swappable which allows plugging in new boards without rebooting the crate. The EPICS IOC needs to support such features, which means, loading the device drivers for the new boards without rebooting the IOC software.

The existing EPICS base does not support dynamic record loading. All records must be loaded before executing the iocInit() command. In order to support loading records after iocInit(), the EPICS base needs to be extended.

EPICS Base Extension

Figure 3 shows the architecture of part of the EPICS base concern to the records.

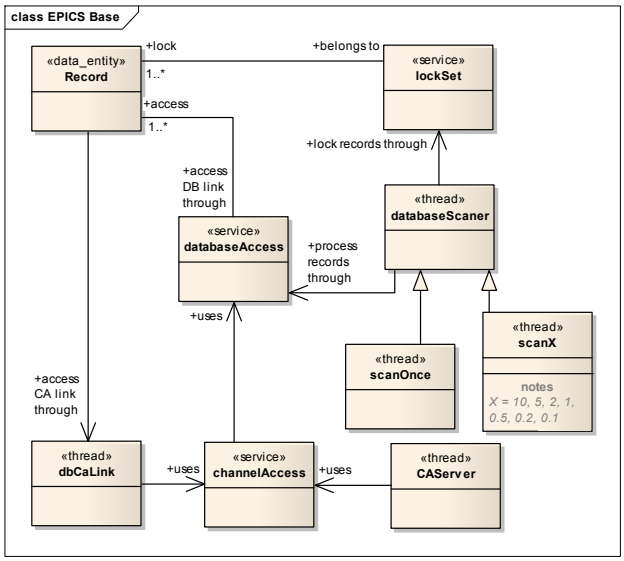


Figure 3: EPICS base modules acting on the records.

Several challenges need to be resolved to load a record during run time:

- Setup the locksets for the newly loaded records. If the new records have no links, simply create new lockset for each of them. If the new records have links with existing records within the same lockset, add the new records to the lockset. And if the new records have links with existing records within different locksets, merge the locksets and add the new records to it.
- Convert PV links to DB link or CA link during record initialization. When converting the PV links, the destination records may have not been loaded yet, so, when a new record is loaded, the record instances loaded previously need to be reconsidered to finish the link conversion.
- Modify the links of existing records. When a new record is loaded, the existing records may want to change their link to point to the new record. In this case, the lockset of the record to be modified should be split firstly and then merged with the lockset of the new link destination.

For solving the problems listed above, a new source file (dbRecordDynamic.c) is added to the EPICS base [3].

The OOEPICS framework and the dynamic records loading functions have been successfully tested in a soft IOC controlling a Newport motion control stage.

CONCLUSION

The OOEPICS framework provides a way to fully access the EPICS records from the user code. The object oriented technology can directly map to EPICS design. It also provides a common way to create, initialize and setup the device driver from the IOC shell. It also enables the user to do EPICS development without knowing much of EPICS, avoiding the long learning curve of EPICS.

The dynamic records loading development provides a good support for hot-swappable system control which enables a complete EPICS based solution for MicroTCA system in the following projects at SLAC.

REFERENCES

[1] http://www.uml.org/
[2] http://www.omg.org/mda/
[3] https://blueprints.launchpad.net/epics-base/+spec/dynamic-record-loading

ISAC EPICS ON LINUX: THE MARCH OF THE PENGUINS

J. Richards, R. Nussbaumer, S. Rapaz, and G. Waters,
TRIUMF, Vancouver, B.C., Canada

Abstract

The DC linear accelerators of the ISAC radioactive beam facility at TRIUMF do not impose rigorous timing constraints on the control system. Therefore a real-time operating system is not essential for device control. The ISAC Control System is completing a move to the use of the Linux operating system for hosting all EPICS IOCs. The IOC platforms include GE-Fanuc VME based CPUs for control of most optics and diagnostics, rack mounted servers for supervising PLCs, small desktop PCs for GPIB and RS232 instruments, as well as embedded ARM processors controlling CAN-bus devices that provide a suitcase sized control system. This article focuses on the experience of creating a customized Linux distribution for front-end IOC deployment. Rationale, a roadmap of the process, and efficiency advantages in personnel training and system management realized by using a single OS will be discussed.

INTRODUCTION - HISTORY

The ISAC control system, implemented using the EPICS toolkit, is completing a migration to use only the Linux operating system for both channel access clients and IOCS. This differs significantly from the original system concept of 1997 in which all IOCS were Motorola VME CPUs running VxWorks, SUN/Solaris was used for the development and production servers, and operator console displays ran on Windows PCs acting as X-terminals [1].

To solve reliability problems, Redhat Linux desktop distributions replaced Windows as the operating system on the operator consoles as early as 2003. Also in 2003, the first i486 processors in the PC104 form-factor and running VxWorks were deployed as IOCs that interfaced with PLCs using TCP/IP [2]. VxWorks boot parameters were installed into the PC104 BIOS via the JTAG interface, which required legacy parallel port hardware and Windows software. To replace or re-function a unit required a new JTAG download via a dongle. By 2005, more powerful Pentium III-based VME CPUs (GMS Mariner – 64 MB RAM) running VxWorks were introduced as replacements for selected MV162 (16 MB RAM) installations.

At that time it became apparent that the maintenance and knowledge support for three different operating systems (Windows,Solaris,VxWorks) was putting too much strain on the resources of the small controls group. As EPICS provided operating system independence for IOCs with release 3.14, a long-term goal was formulated to use Linux as the only operating system in the ISAC control system. Linux is feasible because ISAC is essentially a DC machine which does not demand hard real-time performance from the control system.

Experience using Linux as the operating system on servers, consoles, and developer desktops proved that this single operating system could meet most control needs. Over the period 2004-2010 all the SUN/Solaris hardware was retired and replaced with rack mounted servers running Scientific Linux. Virtualization technology on servers allowed services to be split into single-function servers (database, web, file, dns, ldap) with precise access control. Linux also reduced annual licensing and hardware costs.

The final frontier for Linux deployment was the IOC. Problems with instability on the PC104's VxWorks 5.4 network stack led to replacement with rack-mounted servers running Scientific Linux (SL) which host multiple IOCs. This architecture/OS model met the IOC criteria for stability, reliability, and extensibility. Deployment of single-purpose IOCs on PCs running Redhat desktop distributions began in 2005 [3]. These Linux PCs were used to both develop the IOC application and to deploy it in the field. This "unclean" combination of development and production machine led to the necessity of having "hot spares" available. The PC towers were also prone to hardware failures of hard drives, fans, as well as other problems encountered in the sub-optimal accelerator-laboratory floor environment. As the number of these PC installations grew, so did the effort of Linux administration.

A PROTOTYPE IOC RUNNING LINUX

The need for a small form-factor, low power, no-moving-parts IOC host was identified, and the TS-7350 [*] ARM-based Single Board Computer (PC104 form factor) was chosen as an evaluation platform. Methods for performing software development and field deployment were developed, tested, and deemed acceptable

A project that required a small and highly portable control system for 4 power supplies and 2 vacuum gauges required a microprocessor that was robust, with no fans or spinning media. The ARM units met these criteria. Technologic Systems provide a development kit with a customized Debian Linux distribution and proprietary drivers. A daughter board, TS-CAN1, enabled connectivity to a CAN-bus network using the open-source Ocera-lincan [†] driver, and in-house EPICS device support modelled after existing CAN-bus support. The result was a very small system, with the ARM IOC booting from SD flash within 40 seconds, connected to 5 CAN-bus-attached microcontrollers. A laptop hosting EPICS

[*] http://www.embeddedarm.com/products/board-detail.php?product=TS-7350

[†] http://www.ocera.org/download/components/WP7/lincan-0.3.3.html

channel access extensions serves as an operator console and boot host. This “Control System in a Suitcase” could be deployed for acceptance tests on factory floors. It provided the background experience for identifying the way we want all fieldbus-connected Linux IOCs to work.

EMBEDDED X86 IOC REQUIREMENTS

From the experience gained with the installations mentioned above the following requirements for a generic embedded Linux-x86 IOC were defined (rationale in parentheses):

- No moving parts (increase reliability)
- Network boot – no dongles (simplify maintenance)
  - Reboot time < 40 s (derived from ARM)
- Application agnostic hardware. The target computer does not associate with an application until booted. (eliminates a dedicated "hot spare" per IOC)
- A Linux distribution tailored to ISAC Controls IOC needs (i.e. **Isaclinux**).
  - known kernel composition with all functionality understood
  - kernel to include only services required for our IOC objectives
  - must use standard kernel source distributions
  - root file system must be minimal in size, exposing minimal utilities (reduce security risks, fast boot time)
- Replace the functionality of currently installed VxWorks IOCs
- Separation between application development and production environment – analogous to the VxWorks model where products are created by a cross-compiler on a development host and deployed to different target at run time

METHODS

- The hardware selection for the VME bus was VMIC 7648(Pentium III, 1260 MHz, 512K cache, 128 MB RAM) which we acquired at discount pricing. For RS232/RS485 and GPIB buses we selected a mini ITX motherboard (Atom N270, 1.6 GHz, 512 K cache, 1 GB RAM). These units have low power consumption, low heat load and solid state drives which should translate to high reliability. The ITX provides 2 exposed serial ports, plus an addition 4 on the motherboard. It also provides 2 PCI slots for expansion cards for such buses as GPIB and CAN. This product was considered to have a reasonably long product future and hence a high availability of supply.

- Both hardware architectures selected allow for network boot via their BIOS setups. The industry standard Preboot Execution Environment (PXE) protocol is implemented with pxelinux[‡] and TFTP for diskless, network-only, booting, The VMIC boot time averages 36 seconds. The ITX/Atom pxelinux boot is much slower, 140 seconds. However, by using the enhanced utility, gpxelinux, the ITX/Atom boots in approximately 40.
- Agnosticism was achieved by using a centralized boot system and common root file system. Kernel command line arguments are exposed to the /proc file system and can be accessed by user space applications. A single user space script file "/etc/init.d/rcS" associates kernel command line keywords with functionality. This script is easy to customize, easy to understand, and well structured. Custom keywords are used to identify scripts to run prior to launching an IOC application, such as GPIB or Ethernet private network setups. Other keywords identify the IOC application boot directory and start-up script.
- Customization of an existing Scientific Linux distribution was attempted but abandoned when the difficulties of obtaining kernel sources was apparent. Rather than paring down a bloated desktop release, a very compact Linux distribution was selected as a starting point. Micro Core Linux[§] is a 6MB text-mode distribution of Tiny Core Linux, boots very quickly, uses a Linux 2.6 kernel, and utilizes Busybox[**] for core utilities. The entire Micro Core Linux file system was converted, in-house, to an initrd format. On-going customization of the initrd as well as tools and methods for testing and deployment have rendered the existing incarnation fully disconnected from its ancestry.
  The Micro Core kernel was further pared from 2.2 to 1.8 MB and rebranded as the Isaclinux kernel. Root file system customizations have added 4 MB to the Micro Core starting size, resulting in an Isaclinux file system of 10 MB. The main customization was to the rcS script for launching applications. Other additions for Isaclinux include: ssh server (Dropbear), GNU screen (used to provide a sharable IOC console), C-Kermit (useful troubleshooting tool for serial I/O), EPICS libraries and utilities (e.g. camonitor, caRepeater), and dependency libraries (e.g. libreadline, libncurses, libpcre).
- EPICS OSI library routines for VME are only available for VxWorks and RTEMS. Thus, a

[‡] http://www.syslinux.org
[§] http://distro.ibiblio.org/tinycorelinux
[**] http://busybox.net/

port of a number of EPICS drivers from VxWorks to Linux was undertaken. For the VMIC7648 IOC target, the GE-Fanuc VME_UNIVERSE kernel driver is available[††]. Using the driver's API library a support library for EPICS applications was created that includes a set of access routines that create access windows to the VME bus and convert bus addresses to local addresses for specific VME devices (e.g. sysMapVmeMaster and sysBusToLocalAdrs), interrupt connect routines, and OSI wrappers for VME access. The use of OSI wrappers minimized the software changes required when porting VxWorks device drivers to Linux. Since the VME bus is a Big-endian bus and x86 CPUs such as the VMIC7648 are Little-endian, it was necessary to add byte swapping macros to all VME access operations.

A number of VME device support software changes were also required. For the CAN-bus TEWS Tip810 driver connected via TVME200 VME-bus industry pack carriers, the message receive task was throttled to prevent CPU hogging. The OMS58 receives commands via a 256 byte circular input buffer. On the Pentium based VMIC7648, command acceptance time-outs made it necessary to throttle the send_mess() function in the OMS58 driver code using the POSIX standard function nanoSleep().

- Separation of development and production environments is clearly met by the Isaclinux deployment model. The system is designed to run from a RAM copy created at boot time. Besides being fast, this protects system files from changes and ensures a pristine system on every reboot. This is analogous to the VxWorks model where the VxWorks OS is loaded over the network at boot time. Isaclinux changes can only be introduced via the initrd file system produced on the development server. A Debian development host having the same kernel vintage (2.6.33) as Isaclinux is used to produce the EPICS applications. Using the same C Library and compiler ensures compatibility between run-time environment and EPICS applications

RESULTS – LESSONS LEARNED

One of the primary goals for the Isaclinux project was to standardize deployed systems and stop the proliferation of different hardware architectures, operating system distributions and versions, thereby minimizing system administration overhead. This has been achieved. The development process also offered an opportunity for all authors to become familiar with the internals of the Linux operating system.

The need to customize kernels led to the conviction that a Debian distribution provides a better development platform for embedded systems than Redhat type distributions since access to kernel sources is more standard. The Debian project has official support for embedded systems, including support for the ARM microprocessor. Debian distributions include more up-to-date software related to software development (compilers, libraries, etc.). On the rack-mounted IOC servers that require only TCP/IP to connect to PLCsScientific Linux is still used as the operating system.

We still have some dependency between what we build and where we build it – that is, we have not achieved a cross-native compiling. It would be nice to have a toolchain that creates the target kernel, libraries and apps that could be deployed on any Linux server but our project time ran out. Further enhancements that we are developing include a deployment infrastructure. This consists of packaging the root file system in initrd, version control (legacy/stable/test) of each kernel and root file system, benchmark and regression testing on development hardware and a standardized deployment procedure for production. A configuration tool that provides for event logging of changes to dhcp, pxe, kernel, and initrd is under development.

The VMIC deployments have some issues with certain hardware models. Joerger VSC16 scalers occasionally generate “uninitialized interrupt messages”. VxWorks handles these gracefully and performance is not affected. However on Linux these result in the scaler locking on vector 255. A non-elegant stop gap measure that limits us to one scaler only in any VME crate is to always use vector 255. The CAN-bus interrupt handler works well on 1260 MHz VMICs but blocks erratically on 928.1 MHz clock models.

To date we have Isaclinux deployed in a single VME crate and are awaiting a shutdown period to allow replacement of more VxWorks installations. A number of ITX/Atoms that use StreamDevice for GPIB, RS485, and Ethernet fieldbuses have been installed in the control system. We are evaluating their suitability as digital power supply controllers. For the Atoms, we have also implemented Wake-On-LAN to allow remote restart.

The utility of the ARM with its smaller form-factor is still being explored. Although it is not network-boot enabled (uses boot flash), it loads its applications from an NFS mount. We also have found it to serve well as a serial console for other IOCs, and as a CAN-bus loop packet sniffer.

CONCLUSIONS

The ISAC Control System has a response rate requirement of only 10 Hz. This can be met by an operating system such as GNU/Linux that is not classified as real-time. To minimize IOC application pre-emptions, it is advisable to have the IOC Linux host run only essential services. Isaclinux is a small footprint,

[††] Customizations by TRIUMF DAQ Group.

customized version of a 2.6 Linux distribution. It provides minimal services such as SSH, NTP, DHCP, and NFS needed for a network booted IOC. Its use on such platforms as the GE-Fanuc and Atom allow control of devices on fieldbuses such as VME, GPIB, RS232, and Ethernet.

Linux is now used in all tiers of the ISAC Controls task model including servers, operator consoles, and IOCs. One operating system, understood by all controls personnel, allows a small group of control engineers to share maintenance and development duties easily. This strategy facilitates a sharing of administration duties, concentrates on one platform only for device drivers, and has already proven invaluable in troubleshooting situations.

REFERENCES

[1] R. Keitel, M. Leross and G. Waters, "Control system prototype for the ISAC radioactive beam facility," ICALEPCS97, Beijing, 1997

[2] G. Waters and R. Nussbaumer, "TRIUMF/ISAC EPICS iocs using PC104 platform," ICALEPCS03, Gyeongjiu, 2003

[3] R. Nussbaumer and G. Waters. "BACnet support for EPICS," ICALEPCS05, Geneva, 2005

STATE MACHINE FRAMEWORK AND ITS USE FOR DRIVING LHC OPERATIONAL STATES

M. Misiowiec, V. Baggiolini, M. Solfaroli Camillocci, CERN, Geneva, Switzerland

Abstract

The LHC follows a complex operational cycle with 12 major phases that include equipment tests, preparation, beam injection, ramping and squeezing, finally followed by the physics phase. This cycle is modelled and enforced with a state machine, whereby each operational phase is represented by a state. On each transition, before entering the next state, a series of conditions is verified to make sure the LHC is ready to move on. The State Machine framework was developed to cater for building independent or embedded state machines. They safely drive between the states executing tasks bound to transitions and broadcast related information to interested parties. The framework encourages users to program their own actions. Simple configuration management allows the operators to define and maintain complex models themselves. An emphasis was also put on easy interaction with the remote state machine instances through standard communication protocols. On top of its core functionality, the framework offers a transparent integration with other crucial tools used to operate LHC, such as the LHC Sequencer. LHC Operational States has been in production for a year and was seamlessly adopted by the operators. Further extensions to the framework and its application in operations are under way.

RUNNING LHC

Introduction

A ***state machine*** is a model used in computing. It is composed of states connected by transitions and associated with actions. State machine holds a status where an active state is its most essential attribute.

LHC

During the first period of LHC run it was decided to introduce a state machine with the scope of increasing the operational performance and ensuring control of the LHC operations.

The *LHC Operational States* state machine has a threefold aim of:

- minimizing the risk of errors and mistakes,
- lowering human risk factor,
- increasing machine efficiency by reducing time losses and creating more rigid structure for the LHC nominal cycle.

Meeting such goals has a natural consequence in form of diminished flexibility for the operations, the trade-off all have agreed upon. Moreover, although *Operational States* helps maintaining a high level of equipment safety, it is not a safety but an operational tool.

In order to have a reliable convenient tool, able to cope with the large number of different situations faced by the LHC operations, the state machine had to be carefully studied and prepared. Twelve states were identified, as shown in Table 1. Each LHC machine state is a condition in which the LHC can be found during its lifecycle. In any state, the execution of certain tasks has been restricted (i.e. while the state machine is in "preparation" and there is no beam it is not possible to declare "stable beams" mode) to ensure that no wrong action is performed in critical operational phases.

Table 1: Description of LHC Operational States

state	description
PREPARATION	LHC is prepared to inject beam. All devices are prepared, magnets set to injection current.
INJECTION PROBE	The probe beam is injected and the beam parameters (tune, chromaticity, etc) are checked before injecting nominal beam.
FILLING	Nominal bunches are injected filling the machine.
PREPARE RAMP	Orbit correction and device preparation for ramp is performed.
RAMP	Energy ramp from injection (450 GeV) to nominal.
SQUEEZE	Squeeze beam to nominal value.
ADJUST	Beams are set in collision.
STABLE BEAMS	Experiments can move in their reading devices.
BEAM DUMP	A programmable or unplanned beam dump has happened.
TEST	Used during Stop to test software changes. No limitations applied.
STOP	LHC is not operational (cryo stop, device repair, technical stop).
INJECT DUMP	Special test with operational configuration: beams are injected then dumped immediately after.

STATE MACHINES

Basic Terms

State machine diagram, a ***layout***, is composed of ***states*** and ***transitions***. Each transition is directed, linking precisely two states, hence the whole diagram constitutes a directed cyclic graph. States and transitions are uniquely named, as well as the state machine itself, the ***instance***. Exactly one state is qualified ***initial***, while the others can be assumed ***final***, from which nothing else can be

reached. The current state of the instance is marked ***active***. In principle, there may be only one initial and one active state, being the same at the start of the instance lifetime.

Moving from state A to state B is feasible only if A is *active* and there exists a *transition* from A to B. Moreover, a set of ***actions*** can be assigned to the layout and required to be successfully performed while moving between states. Performing an *action* may relate to any additional operation executed by an *instance* and yield a success or failure. If an action is identified ***fail-forward***, failure is considered neutral to the *move*, otherwise the new *active* state must be decided.

An *action* can be placed in one of three ***locations*** with regard to a single transition or state:

Table 2: Location of Actions

(1)	***on exit***	while *leaving* state A; all transitions from A
(2)	***on transition***	while *moving* from A to B; transition A->B
(3)	***on entry***	while *entering* state B; all transition towards B

Assigning action locations helps to organize the layout and order the actions on the transitions. The actions are performed in natural order of sets: (1), (2) and (3). Those sets can be further sorted if necessary, so that it is known prior to the move in which order actions are called.

Actions are logically divided into two types: ***tasks*** and ***conditions***. *Task* is an operation which holds a significant impact on the environment, changing the state of external entities. An example would be an archiver, logging information to the database, or a publisher which broadcasts messages upon calling. Performing the *task* may end up in failure, although it should not be considered harmful to the *move*. Conversely, *condition* has little impact on the environment, yet it checks its particular quality. Calling the condition results in a boolean value, but should bring no side effects. Failing condition check should inhibit the *move*.

Concepts Omitted

Due to the complexity of the notion, *State Machine Framework* does not support any action rollback mechanism. Each action as a separate entity, unrelated to others, should clean itself up in case of its failure.

Some other reoccurring state machine concepts have proven superfluous in our environment. Hence lack of support for sub-states, nested states, parallel transitions or synchronization blocks that can be found on the state machine library market [2].

LHC OPERATIONAL STATES

State Machine Instance

LHC Operational States is a production instance of *State Machine Framework* based on the specific LHC configuration and dedicated action implementations. The instance follows classic 3-tier architecture with a clear separation of concerns. The middle-tier server runs on a Linux server in CERN computing centre. Configured with twelve states, it manages walking through over twenty transitions. Each transition is equipped with a series of checks that need to be satisfied in order to successfully move forward. They are triggered by a dedicated state machine action, but performed on a remote machine as Sequencer tasks. The results of the checks are communicated back to the core of *Operational States* server where the decisions about the new active state are taken and published further to interested parties. Those being all the listening clients, the database storage and local logging files. The primary clients are Sequencer GUIs, State Machine GUIs and any application using SM client API, although only the first two are authorized to perform state changes.

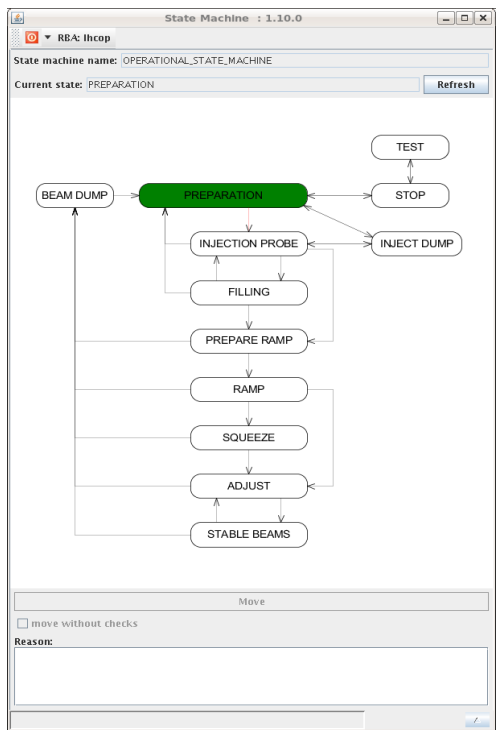


Figure 1: LHC Operational States diagram in SM GUI.

Sequencer

LHC Sequencer [3] is a crucial operators tool. It allows for executing preconfigured, ordered series of operations, i.e. *sequences*. They exemplify the steps taken in order to move the machine along its operational lifecycle. Thus, Sequencer naturally employs the state machine concepts, becoming both its client and executor. Each transition of *Operational States* is requested from a certain Sequencer task. *Operational States* server reacts to the request, among others, by executing its action which in turn launches another Sequencer task.

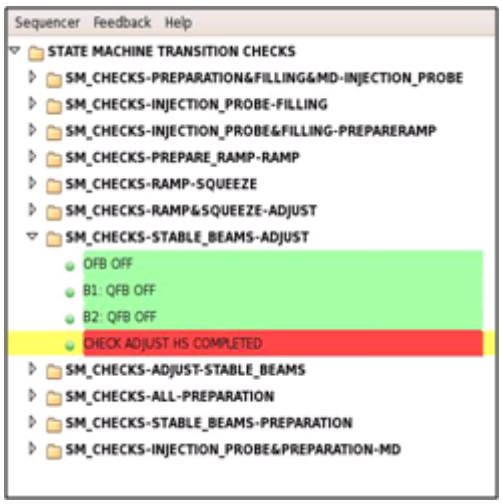


Figure 2: Sequencer checks for transition in SM Checklist.

Those tasks (checks) examine the actual context of the LHC machine, e.g. the external subsystems must meet certain conditions to safely allow to progress to the next stage. In case the check is successful, new active state is decided and returned back to the Sequencer. Otherwise, the operators must take necessary measures so that the conditions are met while the request is repeated.

SM GUI and SM Checklist

To help the operators in visualizing the states diagram two graphical applications were developed. *State Machine GUI* presents the layout indicating the active state, the undergoing transition and the messages issued from the Operational States server. It lets test the transitions outside the Sequencer tasks to ease up their preparation. It also permits to skip certain checks. Such tool proves indispensable in the early development or debugging stages. *State Machine Checklist* is an easy means of presenting the Sequencer checks associated with the transitions. Both tools are extensions to the LHC Sequencer, used on a daily basis in CERN Control Centre.

STATE MACHINE FRAMEWORK

Library

State Machine Framework (SMF) as most of CERN accelerator controls software is created in Java using the Spring framework [4]. Communication between client and server layers of the classic 3-tier architecture is realized with RMI and JMS protocols, employing high level concepts and infrastructure of JAPC [5]. Applications connect to the server through a dedicated client API that allows for performing state changes in both, synchronous and asynchronous ways. In addition to sending the response to the transition requests, server publishes the results through asynchronous channels of JMS. Any client application interested in following state changes, including failed attempts, can become a listener.

SM instance is identified by a unique name that is announced along with the addressing details to the Directory Service [6]. Client library looks up the Directory Service to discover that information. Using an intermediate publishing service greatly improves the flexibility of the SM instance deployment.

By default, SM instance stores its lifecycle data, including state changes, in several resources. Database archive and local log files are the major ones. The current status of the state machine is always preserved between the restarts of the server which provides for smooth upgrades.

Configuration

Configuration of a SM instance can be supplied either through an XML file or a database. In each case, template schema is provided to verify the syntax. Seemingly easier way is to define the layout as an XML document, more readable and simpler to edit. That was also the choice for LHC Operational States, letting operators manage the configuration fully on their own. It has proven an excellent opportunity for stimulating the ultimate users to actively participate in creation and maintenance of the software.

Deployment

SMF source code is split into several packages. Client and server libraries are clearly divided, while actions are shipped in another package. It lets action developers, often outside the state machine team, to contribute their implementations without interfering with main course of *SMF* development. It also alleviates the testing process.

Actions package is combined at the level of deployment of the state machine instance. Lacking an action class results in calling a default one, hence does not break a running server. SM instances are always bundled into a separate product, loosely coupled with the framework packages.

Actions Interface

Actions are generally developed to the dedicated API outside the *SMF* core. They are placed in the instance layout simply through editing its XML/DB configuration. There it is decided if an action falls into a task or a condition logical category. In the action main call, *perform()* method, a variety of information is provided by the SM server, including remote client token and a full SM context from the time of the call. In case of a problem it can throw an exception indicating a non-recoverable error. Otherwise, the action is considered to have completed successfully.

Request Lifecycle

SM instance accepts client calls (RMI) and based on the supplied arguments decides if a state change is possible at the moment. The feasibility of the move is also verified against the SM layout before the call is executed, at the client level. Each request holds additional data, a *payload* that identifies the client application. It can also contain a user provided map of properties. Payload is made available to each action performed on the requested transition, thus client data can be processed at that stage.

Once verified, the request is translated into a series of conditions and tasks that are performed on the transition. The result is then returned to the client, published asynchronously and persisted. Exception or warning messages thrown by the actions are disseminated too. While a single request is handled, all the other incoming requests are put on hold by the server.

Concurrency

One of the key benefits of *SMF* is the soundness of the concurrent model employed. State machines open to a parallel use of many clients are prone to a variety of errors, typically hard to detect at the testing stage. Hence a diligent effort was taken to design and implement a flawless concurrency engine based on Java monitors. As clients are rightly handled also in case of heightened number of requests, the *SMF* proves an accurate solution for fast-changing environments. Over a year of intensive use of *SMF* has confirmed a bug-free environment in that respect too.

Safety

Unlimited access to the SM instance could pose a serious threat to the LHC environment, even if the Sequencer checks have no impact on the probed environment. Role based access model *(RBAC)* was employed to handle the issue, a common solution in CERN controls systems [7]. Each client request's payload is equipped with RBAC token that holds verified credentials of the user. Server decides upon its contents whether the request can be authorized. In case of *LHC Operational States* the access to the SM instance is narrowed solely to the LHC operators.

Embedded Mode

SMF supports not only 3-tier architecture with its separation of clients, server and resources, but also an embedded use within another application. The instance is then accommodated into the scope of controlling application using Spring context, whereby all the client calls are executed locally, between Java threads. Embedded mode is achieved by an appropriate configuration of the instance. No remote calls or auxiliary resources, e.g. Directory Service, are required.

Considering its simplicity and concurrency soundness, such a solution is advised in case a state machine is needed within a more complex application.

CONCLUSIONS

State Machine Framework and its first instance have been operationally used for almost a year now. Over the period they have become an integral constituent of LHC controls software architecture, running impeccably throughout. LHC Operational States instance has reliably intertwined with the existing infrastructure, whereby the LHC Sequencer plays the essential part. The collaboration with other software teams helped to improve the functionality *SMF* offers, while the feedback of the LHC operators have been continuously integrated. Open architecture promoted their active participation in the development of the library. It has also raised several subjects that are being considered for new requirements, thus more work is anticipated in the near future.

State Machine Framework is a general purpose library aimed at both standalone or embedded use wherever state machine concepts need putting in place. Minimal dependency on the accelerator controls infrastructure makes it a comfortable choice for any project seeking a similar tool.

REFERENCES

[1] R. Alemany-Fernandez and M. Lamont and S. Page, “LHC Modes”, CERN EDMS, LHC-OP-ES-0005, 2007
[2] W3C SCXML, http:///www.w3.org/TR/scxml
[3] V. Baggiolini and R. Alemany-Fernandez and R. Gorbonosov and D. Khasbulatov and M. Lamont, “A Sequencer for LHC Era”, Proceedings of ICALEPCS’2009, Kobe, Japan.
[4] Spring Framework, http://springframework.org
[5] V. Baggiolini et al, “JAPC - the Java API for Parameter Control”, Proceedings of ICALEPCS’2005, Geneva, Switzerland
[6] M. Sobczak, “Specification for the Middleware Directory/Name Server”, CERN, 2009
[7] S. Gysin, “Role-Based Access Control for the Accelerator Control System at CERN”, Proceedings of ICALEPCS’2007, Knoxville, USA.

UNICOS EVOLUTION: CPC VERSION 6

E. Blanco Vinuela, J.M. Beckers, B. Bradu, Ph. Durand, B. Fernandez Adiego,
S. Izquierdo Rosas, A. Merezhin, J. Ortola Vidal, J. Rochez, D. Willeman
CERN, Geneva, Switzerland

Abstract

The UNICOS (UNified Industrial COntrol System) framework was created back in 1998. Since then a noticeable number of applications in different domains have used this framework. Furthermore UNICOS has been formalized and its supervision layer has been reused in other kinds of applications (e.g. monitoring or supervisory tasks) where a control layer is not necessarily UNICOS oriented. The process control package has been reformulated as the UNICOS CPC package (Continuous Process Control) and a reengineering process has been followed. The drive behind these noticeable changes was (1) being able to upgrade to the new more performing IT technologies in the automatic code generation, (2) being flexible enough to create new additional device types to cope with other needs (e.g. Vacuum or Cooling and Ventilation applications) without major impact on the framework or the PLC code baselines and (3) enhance the framework with new functionalities (e.g. recipes). This publication addresses the motivation, changes, new functionalities and results obtained. It introduces in an overall view the technologies used and changes followed, emphasizing what has been gained for the developer and the final user. Finally some of the new domains where UNICOS CPC has been used will be illustrated.

INTRODUCTION

UNICOS (UNified Industrial Control System) was born more than a decade ago for the development of the LHC cryogenics control system. Since the first application deployed in mid 2001 the enrichment of functionalities and upgrades followed one after the other [1].

Initially, the UNICOS aim was the automation of an industrial process covering the layers of supervision and control. SCADA (Supervisory Control and Data Acquisition) and PLCs (Programmable Logic Controllers) were respectively the components used in those layers. For the first implementation, PCVue32® as SCADA and Schneider PLCs for the PLC layer were the choices. The addition of the Siemens S7 PLCs and the replacement of the SCADA by PVSS (nowadays called WinCC Open Architecture) was the natural evolution as they became recommended CERN components for an industrial control system.

The UNICOS conceptualization in a framework created not only a well defined set of objects but also: (a) an approach to model a process in units (IEC-61512), (b) a generic environment to develop the application specific control logic and (c) a unified way of operating an industrial plant [2].

During the last years the UNICOS framework has been not only successfully applied to the LHC cryogenics but also to many other applications (e.g. gas flow, collimator temperatures, equipment cooling, etc.) and lately to other different domains like ventilation and vacuum installations [3].

The maturity of the framework has induced an evolution towards even a more reusable and open model where the components provided by the framework are not only restricted to PLCs in the control layer [4]. In this way, UNICOS has been also applied to supervision and monitoring applications as the QPS (LHC Quench protection systems), SURVEY (alignment of the focusing LHC low beta magnets) or the PIC (Powering Interlocks controllers) among others.

The process control package has been reformulated as the UNICOS-CPC package (Continuous Process Control) and a sound reengineering process has been followed to finally create the UNICOS-CPC v6.

THE REENGINEERING APPROACH

After more than 10 years of usage of the framework in different domains, process and automation engineers together with plant operators have naturally identified several potential improvements and new requests.

The LHC detectors Gas Control project created additional objects to cope with missing framework functionalities (i.e. parameterization, simple status and analog alarms) which were not initially included in the basic package.

One of the framework features is the use of tools to automatically generate the code; those tools (based on Microsoft Access) although very valuable were unnecessarily rigid for the creation of new object types and rather inefficient when dealing with large projects.

The application dependent control logic is composed of a set of templates that combine together PLC code and a meta-language to make use of advanced features during the code generation (e.g. code reusability, user parameterization, etc.). Users regularly reported difficulties in understanding those logic templates.

All these facts were the driving force behind the vigorous reengineering process of the UNICOS-CPC package.

NEW FEATURES

Some of the package enhancements are: (1) new improved functionalities in almost all its objects as well as new objects, (2) extended flexibility to ease the creation of new objects, (3) the new tooling suite for the automatic generation of the applications, (4) improvement readability and usage of logic templates, (5) local operation enabled, (6) the change of the Siemens PLC

application architecture, (7) dynamic recipes and, finally (8) HMI improvements.

Basic objects improvements

There has been a rationalisation of the existing CPC objects trying to keep their generic characteristic yet. This task has ended by the removal of redundant objects (e.g. Digital Input Calculated is integrated in the Digital Input), deletion of non used objects (e.g. Computed) and creation of new objects (e.g. AnaDO). AnaDO holds an Analog and Onoff objects hybrid behaviour and it is found in most of the industrial pumps in fields like vacuum, cooling and HVAC.

A detailed list of the basic objects together with their category is presented in the Table 1.

Table 1: UNICOS-CPC Basic Objects

Name	Function	Category
xI	x=Analog/Digital Input	I/O
xO	x=Analog/Digital Output	I/O
AIR	Analog Input Real	I/O
AOR	Analog Ouput Real	I/O
xPar	x=Analog/Word/Digital parameter	Interface
xStatus	x=Analog/Word/Digital status	Interface
Analog	Valves, Heaters,...	Field
OnOff	Motors, Pumps,...	Field
AnaDig	PWM, slide valves	Field
Local	Local hand valves	Field
AnaDO	Pumps, Frequency Variators	Field
xAlarm	x=Analog/Digital alarm	Control
Controller	PID feedback control	Control
PCO	Process Control Unit	Control

Among all the modifications in the objects, important changes to mention are:

- Full stop functionality in the field objects. Up to now the PCOs could restart once the interlock was acknowledged and their cause had disappeared. In the new version, the control engineer may specify that the device in full stop mode must be explicitly allowed to restart by the operator. This decouples the interlock acknowledge from the operator explicit restart action.
- Controller mode of operation compliant with the UNICOS modes and standardized working states (regulation, output positioning and tracking). The engineer can work with normalized or with physical units. The cascade controller structures can be now generated from specifications.
- Double inverted output in the OnOff objects.
- Addition of a real PWM (Pulse Width Modulation) functionality inside the AnaDig object.
- Alarms on the Local objects expected positions (e.g. a hand valve expected to be closed but its low end-switch is not active)
- Analog alarms thresholds can be parameterized or imposed by the logic code dynamically.

New Objects Creation by the use of a Meta-model

One of the reengineering process tasks was the polishing of existing objects. There were several inconsistencies between the objects in terms of their internal functionalities, interfaces and naming homogeneity. The proper way to homogenize them was the creation of a UNICOS model.

The model is supported by a meta-model describing the properties of the model [5]. The model describes families which are used by the different objects (e.g. *FEDeviceParameters* to define the object parameters). Having such model has both allowed the refinement of the objects and also allowed the framework to create new objects. To support the creation of the objects a specialized tool was developed: the TCT (Type Creation Tool) which permits a drag-and-drop based mechanism to build new objecs and create the device type definition based on XML (eXtended Markup Language).

UAB: Factory Automation Tools

The previous version of the UNICOS-CPC relied on the automatic generation tools based on Microsft Access. Two different generators were used for each PLC supplier (Schneider and Siemens). The code generated by the tool was somehow attached to the generator itself and the project versioning was difficult.

The importance of appropriate flexible and scalable tools is more visible when dealing with large control systems. Control engineers require them to develop such large systems within a reasonable time. A clear example of this is the LHC cryogenics control where some PLCs are configured using thousands of source code files. Using appropriate factory automation tools optimize the time of application development and increase the engineer performance. Note that one such PLC could deal with more than 2500 I/O channels and 700 field objects (e.g control valves, heaters, etc.).

The need of a more modern and flexible tool dealing with the previous mentioned shortcomings lead to the creation of the UAB (UNICOS Application Builder).

The UAB is formed by different components each one having a specific functionality (See Fig. 1). For example a Siemens S7 plug-in will create the S7 PLC application source code while the SCADA WinCC OA plug-in will be in charge of generating the needed information for the SCADA configuration. A special plug-in, the so-called CPC Wizard, orchestrates the plug-ins involved in creating the control application and shows the control engineer a friendly vision of all the steps to create the process control application [6].

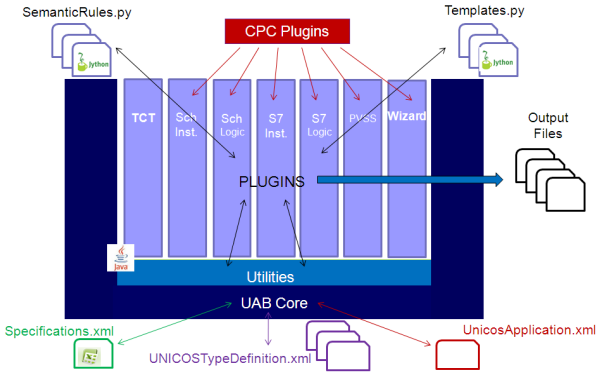


Figure 1: UAB plug-ins.

Another remarkable feature of the UAB architecture is the possibility to include others plug-ins profiting from the already developed core functionalities. This is the case of CoDeSys (Controller Development System), a new plug-in presently being developped which will include a new common IEC-61131-3 based PLC platform.

Control Fogic: Functionality and Femplates

The control logic templates have been rationalized both, by grouping functionalities which were spread out in several templates and also homogenizing them between the two PLC platforms. Table 2 shows the existing standard control logic templates specifying those which are fully generated by the UAB tool.

Table 2: UNICOS-CPC Standard Logic Templates

Name	Function	User edition
BL	Basic logic	No need
CL	Configuration logic	Yes
IL	Interlock logic	Yes
CDOL	Common Dependent Objet logic	No need
GL	Global logic	Yes
DL	Dependent logic	Yes
SL	Sequencer logic (optional)	Yes
TL	Transition logic (optional)	Yes

The format of the logic templates, where the control engineer fills in the process specific logic, has changed. Previously they were text-like files where PLC programming code was explicitly written together with a special language based on Visual Basic to exploit parameterization. Now, with CPC v6, the templates are Python oriented and naturally exploit the benefits of using a scripting language to generate the final PLC code. The advantage: an effortless way to exploit the parameterization included in the control specifications.

Local operation: large vs. small scale projects

Both large and small industrial facilities have different requirements with respect to their automation and operation. When dealing with large projects, the requirements for operation and/or development are quite restrictive and, usually, operation requires centralized control rooms. On the other hand, often, the operation of medium or small projects is not realized 24/7 in front of dedicated HMIs, rather, a subset of information is communicated permanently to a multi-purpose control room where operators, following a multitude of general services, get this information from various processes (e.g. electricity, cooling, ventilation, etc.). Usually that information is a, reduced but still meaningful, set of alarms which should trigger the attention of the operators who, in turn, ask the experts to deal with a problem in case of need. Following this triggering event, the expert connects to the SCADA to check the details of the problem.

There are also cases where domain experts need to command industrial installations locally, and hence, the need of a local interface: the industrial operation touch panels answer that requirement. Availability is also an issue, because the classical SCADA operation relies on Ethernet networks that provide the supervision servers access to the PLCs. Local panels connect naturally to the PLCs with native industrial fieldbuses (e.g. MPI, Profibus, etc.) and exhibit the needed robustness for use in an industrial environment. The new version of UNICOS CPC includes a new mode of operation called *local*. This mode is activated once the object is taken by means of the local panels deployed in the field. The package incorporates a library of objects (faceplates and widgets) to build applications in local touch panels in the same way of what has been offered in the classical SCADA.

Siemens PLC New Architecture

The experience gained with the SIEMENS PLCs in the UNICOS-CPC environment presented a noticeable drawback. In fact, the creation of a new object in the package imposed large modifications to the PLC application architecture. The goals, during the reengineering phase, were to remove this constraint and also to eliminate any required user parameterization once the application is generated.

The usage of the UNICOS model imposed the standardization of the existing and future objects. This, in turn, allows an optimal integration of new objects and an optimal usage of the SIEMENS S7 PLC resources.

Operation using Flexible Recipes

An important feature of a large control system is the ability to apply the appropriate set of parameters at the time they are pertinent. The previous version of the package already included the capability of defining recipes but with a disadvantage: the need of a priori definition of the recipe components.

A flexible recipe mechanism has been developed in the new package version. Flexible recipe management is recognized as a highly desired capability during operation where the plant operators need to tweak the recipes and adapt their components and values to their requirements without rebuilding the control application. The designed recipes can run in multiple PLCs and still be synchronized in its triggering.

The mechanism is PLC platform independent and it is designed in an open and reusable basics, thus becoming in a SCADA component.

HMI Improvements

There have been many modifications concerning visualization in widgets and faceplates: without enumerating the all, there are some new features which deserve attention.

First, because there is a recognized need today to show on the same screens many different control systems (e.g. cooling, HVAC, vacuum, etc.), the colours employed in the HMI had to be redefined. There are no standards in this field, so the engineers must be judicious in their choice. The result is a decrease in the number of employed colours to 4 base colours (green=ok, red=alarm, orange=warning and yellow=force values) and 2 additional error indications (cyan=communication and purple=no data).

Second, the devices can be referred to both with the control name and the expert name. This new feature allows the different experts to have their own project dependant nomenclature in naming their devices.

Third, the alarms representation incorporates the level concept. Following the recommendations of the ANSI/ISA-18.2 standard, a new field on the alarms explicitly characterizes the type of alarm. The four levels employed are fully compatible with the classification of the CERN alarm system.

And four, the objects incorporate two additional navigation options, their positions within the control hierarchy and also the state of the interlocks affecting them in the case of the field and PCO objects.

CONCLUSIONS AND FUTURE PROSPECTS

A large reengineering process has been carried out on the UNICOS CPC package. As a result, CPC v6 provides a better tool suite to automatically create their industrial control applications and also a framework where they can easily extend the objects library in a standard way. The package is already being used in the CERN cooling and HVAC processes.

Following this large reengineering phase, the UNICOS CPC v6 package accepts new capabilities. There are 4 main axes where the tool could develop: auto tuning of controllers, advanced control, redundancy and finally safety systems (see Fig. 2).

Control systems performance depends largely on the correct behaviour of controllers. Most of them are PID (Proportional, Integral and Derivative) and the task of tuning such controllers is widely recognized as a hard job. The package plans to offer automatic tuning mechanisms to the operators.

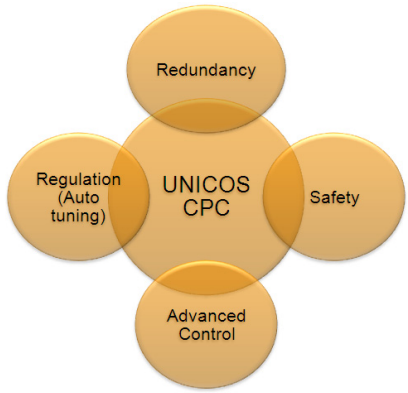


Figure 2: Future work lines.

Some processes show strong nonlinear behaviour and then are intrinsically difficult to regulate. Advanced control solutions are developed to cope with such processes and to help operators. Basic modern control includes a portfolio of solutions from feed-forward, override, cascade to other more sophisticated as Smith predictors and model based predictive control. Some of these solutions are already in the UNICOS CPC v6 (e.g. override, cascade), others will be integrated in the following versions.

Redundancy at the control layer ensures increased availability in the control system. Some processes impose redundancy because of their large downtime in case of PLC failure. UNICOS-CPC will ensure the integration of such components for both Schneider and SIEMENS PLCs.

Finally, safety systems are more often required by the control engineers in processes where safety (personnel and/or equipment) is an issue. Plans for the creation of a similar framework for safety systems are being prepared.

REFERENCES

[1] J. Casas-Cubillos, et al. “Application of Object-Based Industrial Controls for Cryogenics”, EPAC’02

[2] Ph. Gayet, et al. “UNICOS a framework to build industry like Control systems: Principles and methodology”, ICALEPCS’05, Geneve 2005

[3] D. Willeman, et al. “UNICOS CPC New domains of applications: vacuum and Cooling and Ventilation”, ICALEPCS’11, Grenoble, 2011

[4] H. Milcent, et all. UNICOS: An Open Framework. et all. ICALEPCS 09

[5] B. Copy, et al. “Model Oriented Application Generation for Industrial Control Systems”, ICALEPCS’11, Grenoble, 2011

[6] B. Fernandez Adiego, et all. “UNICOS CPC6: Automated Code Generation for Process Control Applications”, ICALEPCS’11, Grenoble, 2011

RULES-BASED ANALYSIS WITH JBOSS DROOLS : ADDING INTELLIGENCE TO AUTOMATION

E. De Ley, D. Jacobs, iSencia Belgium, Ghent, Belgium

Abstract

Rules engines are less-known as software technology than the traditional procedural, object-oriented, scripting or dynamic development languages. This is a pity, as their usage may offer an important enrichment to a development toolbox.

JBoss Drools is an open-source rules engine that can easily be embedded in any Java application. Through an integration in our Passerelle process automation suite, we have been able to provide advanced solutions for intelligent process automation, complex event processing, system monitoring and alarming, automated repair etc.

This platform has been proven for many years as an automated diagnosis and repair engine for Belgium's largest telecom provider, and it is being piloted at Synchrotron Soleil for device monitoring and alarming. After an introduction to rules engines in general and JBoss Drools in particular, we will present its integration in a solution platform, some important principles and a practical use case..

PHYSICAL SCIENCES AND INTELLIGENT PROCESS AUTOMATION

General Vision

In today's world of physical scientific research, the usage of and the dependency on advanced technology has become crucial. As the boundaries of research shift, the requirements on tools for exploration, experimentation and verification become ever more complex and extreme.

The required investments, both in costs and in time, are typically huge and can no longer be duplicated by all parties involved in related research domains. Collaboration in the design and usage of advanced “centralized” or “shared” research infrastructure, like synchrotrons, becomes mandatory.

The institutes offering such infrastructure then effectively become service providers for all the parties involved in related research disciplines, both for internal scientists and for visiting researchers (or groups).

Offering complex infrastructure as a service also implies extensive support organizations and processes, like project approval procedures, safety and security regulations, HR, IT support, resource planning, repair workshops, vendor management etc.

Which in turn implies, maybe a bit paradoxically, that the more advanced and exotic the technological requirements for executing scientific research, the more the organizational aspects get closer to traditional requirements for professional service providers in non-scientific domains. But with a clear long-term advantage, compared to many professional organisations : scientific institutes have a single long-term goal - *offer the best-possible infrastructure and support for advancing science*. This clear goal can drive the complete organisation to operate in a transparent and collaborative way.

It is our vision that a *common open software platform for intelligent process automation* can be of tremendous value to assist an organisation in achieving this goal, across the majority of its activities by removing administrative duplication, reducing risk for errors in each step, removing the need for continuous on-site presence and combining advanced security with consistency and transparency where needed.

Such a platform can be used for supporting, automating and orchestrating a wide range of processes, from traditional workflows (such as project approval) to advanced algorithms for resource allocation (e.g. beamlines, guest rooms, login accounts etc.), from automated machine monitoring of the accelerators and control sequences for beamlines, to offering secured access to experiments' results.

Concretely for Synchrotrons

Automation in synchrotrons is typically associated with device drivers, sensors, control buses and other technical components, driven by scripts and hard-coded programs. But the scope for automation can become much larger by increasing the level of abstraction. This in turn allows adding advanced features like automated analysis, diagnosis and decision management through the integration of e.g. a rules engine, which is the scope of this paper.

AUTOMATED ANALYSIS AND DECISIONS

Some example uses of enriching automated processes with analysis and decision logic are:

- *What to do next?* Complex and/or dynamic routing or filtering logic based on previously obtained information
- *Is what we did OK?* automated analysis or validation of results obtained in previous steps
- *Just wake me up when you need me.* Intelligent monitoring systems, integrated in the same platform to automate basic control processes and business processes
- *Wait a moment, how did we get in this mess?* Correlating acquired data and elementary

analysis results to obtain a final diagnosis on a problem situation
- *Oops, how can we get this resolved?* Automated advises for corrective actions in case of problems
- *Don't bother me any more for this.* Automating simple recurring decision processes

Automated diagnosis or decision management can be decomposed in the following stages :

- Autonomous monitoring or discovery
  Continuously track some important performance indicators and generate an alarm when a value becomes suspect.
- Context-sensitive analysis of raised alarms
  Typically involves looking at many different datasets and measurements.
  Uses thresholding, trending etc. to identify exceptions (e.g. taking your fever)
- Consolidation of obtained information and analysis results towards a diagnosis or decision (e.g. "you've got a simple flue"). This may optionally include advises for problem resolution or repair.

RULE ENGINES

To cater for the above needs, the software system must be able to handle large volumes of data and to apply complex reasoning logic on it. An easy-to-understand reasoning approach uses conditional actions, also known as "if-then" logic. Such constructs are frequently used in traditional programming, but for representing complex context-dependent logic they quickly become a nightmare to understand or maintain since :

- it quickly becomes tricky to correctly represent all combinatorial possibilities of the conditions with nested if-then-else statements
- the logic is spread out across different parts of the code base
- the result becomes more and more fragile with each incremental adaptation

Rule engines are specialized software systems for applying conditional actions (if/then rules) on data. The use of a rule engine brings following benefits :

- business logic, expressed as rules, can be externalized from the application code. The business logic can then be maintained in a centralized way with dedicated tools.
- rules can be defined in human-readable form in text files, spreadsheets etc.
- rules are often defined in a declarative way. Starting from facts that you know are true, the desired outcome or action is defined.
- rule engines are optimized to evaluate large rule bases, find matching conditions and trigger the corresponding rules.

The term "rule engine" is used in different ways. The ones which interest us are also known as "production rule systems". The central part of such a system is an inference engine that is able to match rules against facts or data to infer conclusions which result in new facts or in actions. Through optimized matching algorithms such engines are able to scale to high volumes of rules and facts.

INTRODUCING JBOSS DROOLS

Drools[1] is the leading open-source rules engine written in Java. It was started in 2001 and became a prominent Java rules engine with its 2.0 release. In 2005 the Drools project became part of the JBoss group and since then Drools also became known as JBoss Rules. In 2006 JBoss was acquired by Red Hat. Besides funding the core development team, they provide professional services like support and training. The current stable version is Drools v5.2.

The Drools suite contains several modules that together form their *Business Logic Integration Platform,* of which the following ones are of interest for our purposes :

- Drools Expert : the actual rule engine
- Drools Fusion : an integrated engine extension to support event processing and temporal reasoning
- Drools Planning : automated resource planning

Drools Engine

The core of the Drools suite is an advanced Inference Engine using an improved Rete algorithm[2] for pattern matching, adapted for object oriented systems, and for Java in particular. Rules are stored in the production memory, while facts are maintained in the working memory.

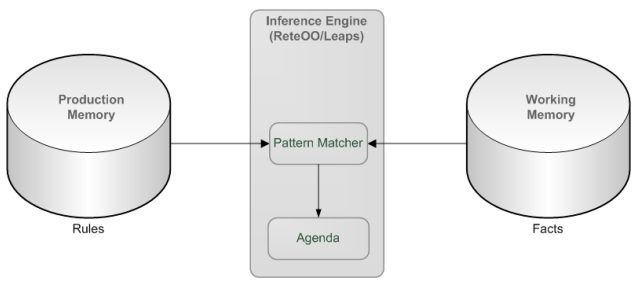


Figure 1 : The main parts of a rule engine.

The production memory remains unchanged during an analysis session, i.e. no rules are added or removed or changed. The contents of the working memory on the other hand can change. Facts may be modified, removed or added, by executing rules or from external sources. After a change in the working memory, the inference engine is triggered and it determines which rules become "true" for the given facts. If there are multiple selected rules, their execution order will be managed via the Agenda, using a conflict resolution strategy.

When a selected rule has been executed, and one or more changes were done in the working memory, the inference engine goes to work again, adapting the agenda and executing the rule that has now reached the highest priority. This iterative process continues until the agenda is empty. Then the analysis session terminates and results can be queried from the working memory or through the usage of global variables.

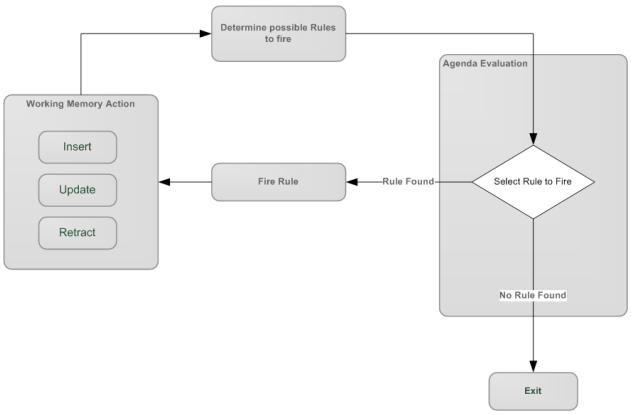


Figure 2 : Iterative inferencing.

Options for Rules Definitions

Drools offers different ways to define rules. The native rules language will feel familiar for Java developers, with the addition of advanced expressions for specifying conditions. For example :

```
rule "Raise prio if 3 pending alarms"
when
    $system : System()
    $alarms : ArrayList( size >= 3 )
      from collect( Alarm( system == $system,
        status == 'pending' ) )
then
    # Raise priority, because $system has
    # 3 or more alarms pending.
   # The pending alarms are in $alarms.
end
```

When repetitive rule patterns occur, one can set-up more readable approaches to define the rules. One way is to create a rule language dedicated to the problem domain, so-called DSL rules.

Rules can then be expressed like :

```
when
  There is a Cheese with
  - age is less than 42
  - type is 'stilton
then
  Taste It
```

which gets translated at runtime in the technical rule language, based on the defined DSL mapping.

A third interesting approach to define rules is to use spreadsheets. This offers a familiar entry tool for non-IT persons to easily define large rules sets as long as they fit in a-priori defined rule templates.

INTRODUCING PASSERELLE

Within Passerelle [3], all processes are defined in graphical models that are executable by one of the available executors. In "real" production environments, the Passerelle Manager, our web-based process automation server platform[4], is the preferred execution engine. It offers a complete solution for process definition, maintenance, execution and follow-up.

In the Passerelle process engine, a process (a.k.a. sequence) is defined by a graphical assembly of actors that each perform a step of the complete process. Assemblies are stored in XML files. Actors get single well-defined responsibilities and are unaware of their surrounding "colleagues". They are only able to communicate via messages sent across channels between the sender's output port and the receivers' input port(s).This strong decoupling improves reusability and maintainability.

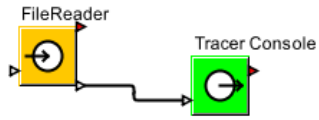


Figure 3 : Reading a file in Passerelle.

In the figure above, the FileReader actor will read a text file from a configurable location, and will send a message for each line to the next actor. In this simple model, this one just dumps the text content of each received message to a Console view.

There are actors available to control Tango devices, to make routing decisions in the process flow, to query databases, send e-mail or SMS etc.

INTEGRATING DROOLS

Drools is easy to integrate in any Java application via :

- Standard JSR-94 Java Rule Engine API
- Drools' own "Knowledge API" with more advanced capabilities, for example to be able to use the Fusion and Planning modules. Passerelle uses the Drools Knowledge API.

One can distinguish the following major API features, required for integration in a production-ready solution:

1. Build a binary version of a knowledgebase packages, from the different types of source
2. Select a knowledgebase package for execution
3. Start an analysis session
4. Feed facts into the working memory
5. Start the inference engine
6. Obtain the results
7. Release session resources when appropriate

Rules and Processes as Versioned Assets

For integration features (1) and (2), Passerelle includes a specialized repository service to maintain *process models*, and *rules assets,* grouped in *project* assets. The service offers a.o. version management, knowledgebase building and packaging, a simple API to retrieve pre-built binary knowledgebase packages and import/export to migrate project assets between test and production environments.

Actors with Intelligence?

Passerelle has specialized actor libraries to integrate the usage of the Drools rule engine in an automated process. These actors utilize the Drools API features (3)-(7) as described above. By configuring the actors with a

reference to the desired rules package, a limited number of actor implementations can be reused in many different processes, each time with the matching rules-based logic.

In this way it is possible to apply rules to analyse the results of the work done in the previous steps of the process, and act on the obtained analysis results, all within a common paradigm of sequences and actors.

Support Tools

Passerelle Manager stores all analysis results, including timed task execution traces, in a relational database. The web UI includes rich views on this data, to aid with application support and with control and improvement of the analysis quality.

EXAMPLE : DIAGNOSIS OF TELECOM LINE - AND SERVICE QUALITY

For the largest Belgian telecommunications provider, a solution has been implemented and is operational since approximately 5 years. To reduce the waiting times for the customers that are contacting the support help-desk, there was a need to be able to automatically trigger a number of tests and measurements on the impacted telephone line and associated services. This should happen even before the customer gets in contact with a help-desk operator, i.e. while still in the waiting queue. Then the operator already has all detailed test results available about the condition of the telephone line, ADSL services etc, when the customer is passed through. Also field technicians, working on-site at their customers can use this automated Diagnosis And Repair Engine.

Each day, approximately 35000 diagnoses are performed for about 700 help-desk operators and 2500 technicians, with dedicated rules for different customer segments, line technologies etc. The result is a dramatic reduction in error rates and a significant increase of “first-time-right” indicators.

EXAMPLE: MONITORING BEAMLINE SOURCE POSITIONS

In this scenario, the goal is to continuously compare beamline source positions to reference values, and to check for slow drift. When the measured position starts drifting, or deviates from the reference position by more than a given threshold, a number of possible root causes must be evaluated (like temperature circuits, insertion device changes etc), and a diagnosis on the plausible cause(s) must be produced.

This requires analysing a stream of position measurement events to :

- compare them against reference values obtained from a snapshot
- check measurement values for drift within a certain time window

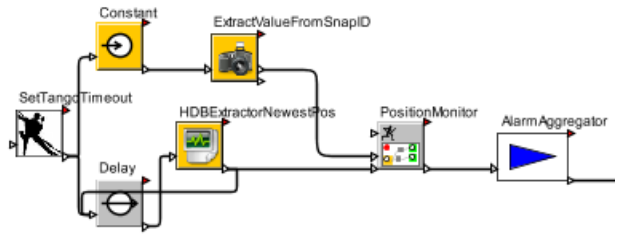


Figure 4 : A fragment of the monitoring sequence.

The monitoring sequence includes an actor to read the reference values from a snapshot and to feed them via an event stream input to the PositionMonitor actor. The HDBExtractorNewestPos actor periodically reads the latest position measurements from the archive and sends these as events to the PositionMonitor. This one can then compare the measurements with the snapshot and also check for drift in measured positions in a time window of e.g. 15 seconds. The below presents an example of using a Java domain model in rules :

```
rule " BLSrcPosition drift > 0.01 within 15 s"
  when
    bl1 : BLSrcPosition($blName1 : beamLine,
        $blPos1 : position )
        from entry-point "BL SrcPos Stream"

    bl2 : BLSrcPosition($blName2 : beamLine,
        beamLine == $blName1,
        this after [0s,15s] bl1,
        eval(position.distanceTo($blPos1)>0.01))
        from entry-point "BL SrcPos Stream"
  then
    // raise an alarm...
end
```

CONCLUSIONS

We have presented the concepts and the added value of integrating a rule engine in a software platform for intelligent process automation. Drools offers many advanced features that are of interest. Besides using standard production rules, it is also possible to perform temporal event-based reasoning and resource planning.

The resulting solution platform can be applied in many different domains, ranging from automated diagnosis and repair in telecommunications customer support to monitoring synchrotron infrastructures to automating beamline experimental sequences.

REFERENCES

[1] JBoss Drools : http://www.jboss.org/drools/
[2] Charles Forgy, "On the efficient implementation of production systems." Ph.D. Thesis, Carnegie-Mellon University, 1979.
[3] Passerelle project information : http://www.isencia.be/services/passerelle and http://code.google.com/a/eclipselabs.org/p/passerelle/
[4] E. De Ley et al. : “Web-based Execution of Graphical Workflows : a Modular Platform for Multifunctional Scientific Process Automation”, ICALEPCS 2011.

INTEGRATING GIGABIT ETHERNET CAMERAS INTO EPICS AT DIAMOND LIGHT SOURCE

T. Cobb, Diamond Light Source, Oxfordshire, UK

Abstract

At Diamond Light Source we have selected Gigabit Ethernet cameras supporting GigE Vision for our new photon beamlines. GigE Vision is an interface standard for high speed Ethernet cameras which encourages interoperability between manufacturers. This paper describes the challenges encountered while integrating GigE Vision cameras from a range of vendors into EPICS[1].

INTRODUCTION

Diamond Light Source is a 3 GeV third-generation light source with a 561 m storage ring (SR), a full-energy booster (BR) and a 100 MeV pre-injector Linac[2]. The photon output is optimised for high brightness from undulators and high flux from multi-pole wigglers. The current operational state includes 19 photon beamlines, with a further three beamlines in an advanced stages of design and construction. A further phase of photon beamlines is now confirmed, detailed design and construction of these 10 beamlines started earlier this year.

A range of cameras are used to provide images for diagnostic purposes in both the accelerator and photon beamlines. The accelerator and existing beamlines use Point Grey Flea and Flea2 Firewire cameras. The disadvantage of Firewire is that it requires complex cabling with multiple repeaters chained together. Diamond has used both a licensed Firewire stack running under vxWorks and an open source Firewire stack running on Linux, but both implementations have been somewhat unreliable. This means that the bus occasionally needs to be power cycled, especially when running at 800Mbit/s over 10m cables.

For the next phase of photon beamlines a better solution was needed, so the decision was made to evaluate Gigabit Ethernet cameras. This would allow up to 100m data transfer lengths, with greatly simplified cabling.

INITIAL TESTS

Selecting a Camera

The most suitable replacement camera was the Prosilica GC1020[3], Fig. 1, as it uses the same sensor as the existing Flea2 cameras, and already had EPICS support in the areaDetector[4] module.

Figure 1: Prosilica GC1020 Camera.

Configuring EPICS Support

The areaDetector module provides a common interface for all supported 2d detectors. Integrating the camera into areaDetector allows image processing and analysis plug-ins to be chained together at runtime, such as the NDStats plug-in for statistics, or the ffmpegServer[5] plug-in to provide a compressed mjpg stream for visualisation (see Fig. 2).

The PvAPI[6] library is supplied by Prosilica (now part of Allied Vision Technologies) to control their cameras in the form of a software development kit (SDK) that works on Windows, Linux and OS X. The areaDetector Prosilica driver consists of a translation layer on top of the PvAPI library.

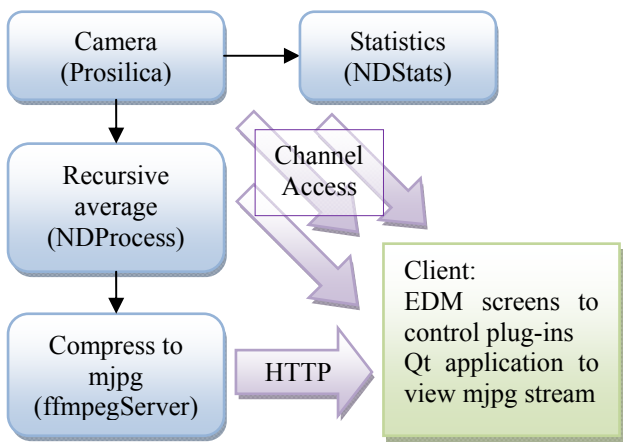


Figure 2: Typical areaDetector plug-in structure.

Conducting a Representative Test

A series of tests was conducted with 10 cameras on a typical beamline network consisting of a Nortel Gigabit Ethernet switch with servers, desktop machines, motor controllers and other Ethernet attached equipment. Different combinations of cameras were tested, and it was noted that the more cameras that were active, the less total bandwidth was achievable without dropped frames. With 3 cameras acquiring, 70MB/s was reliably achievable which was sufficient. Occasionally a camera would hang and stop acquiring, but this was not reproducible later in the lab.

SUPPORTING CAMERAS FROM DIFFERENT VENDORS

All vendors supply an SDK of some form to allow end users to control their cameras, but using one of these SDKs has some disadvantages:

- Most vendors' SDKs are tied to the vendor's cameras, so that you cannot switch to a different vendor without changing SDKs
- Most vendors' SDKs are focussed on the Windows end user, while Diamond uses Linux for control system servers.
- Most vendors' SDKs are closed source, meaning that debugging problems is more difficult, and bugs must be fixed by the vendors.

These factors prompted a search for an open source library that could control any GigE Vision camera, would run on Linux, and could be integrated into EPICS

THE GIGE VISION STANDARD

Most Gigabit Ethernet cameras conform to the GigE Vision[7] standard, which uses the GenICam[8] standard to describe the features supported by the camera. The GenICam standard provides a common interface to many different types of cameras, across different vendors and even across different physical connections types. The camera provides an XML file which describes the features that it supports, and how they map to registers on the device. This means that supporting cameras from different vendors should be simple, as the XML file will describe how the camera should be controlled.

Unfortunately, the licensing conditions of the GigE Vision standard do not allow the licensee to reveal details of the standard. Any licensee wishing to publish software implementing the standard must make the part that is drawn from the standard closed source otherwise they risk breaking the licensing agreement. This would mean that if Diamond used the GigE Vision standard to produce any software implementation, its sources could not be published back to the EPICS community.

In order to be useful to the EPICS community, an open source library was needed, and the only way this library could exist is if it had been reverse engineered. Aravis[9] is one such library.

INTEGRATING ARAVIS INTO EPICS

A typical areaDetector driver consists of a number of parameters, both defined by the base class and specific to this detector, and a mechanism for publishing frames from the detector as NDArrays. As well as the driver, an EPICS database provides access to each of these parameters, and an EDM[10] screen to provide the user interface. The sections below describe the steps taken to wrap the Aravis library in an EPICS module called aravisGigE[11].

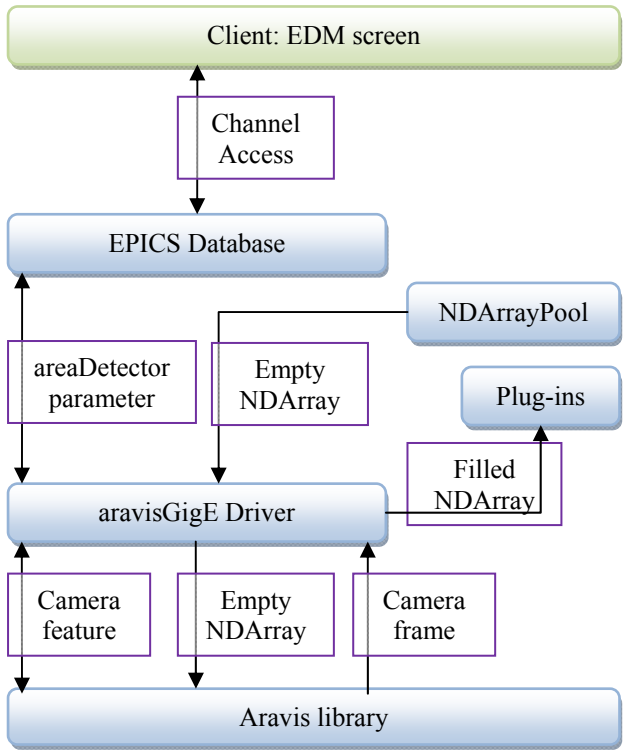


Figure 3: Structure of aravisGigE driver.

The Driver

One of the tasks done by the driver is mapping all the areaDetector common parameters to features supported by the camera. E.g. ADGain is mapped to the "Gain" or "GainRaw" feature. The mapping of areaDetector parameter to camera feature is placed in a hash table. For every other camera feature a new areaDetector parameter is created, and this is also placed in the hash table. When EPICS writes a parameter down to the driver, the corresponding feature is looked up in the hash table, and written to the camera. An update function also periodically updates the readback values of each of the parameters in a similar manner (see Fig. 3).

All memory management of frames in the driver is done using areaDetector's NDArrayPool, which maintain a pool of NDArrays that can be reused when all plug-ins have finished with them. The Aravis library is loaded with a number of NDArrays which it fills in when it gets a new frame.

The driver registers a call-back function with the Aravis library that is called whenever the camera produces a new

frame. This frame is annotated with information like timestamp, colour mode and data type before being published to any connected plug-ins.

Extracting the GenICam XML

Although the driver can query the camera to find out which features it has and create areaDetector parameters on the fly, the EPICS database and EDM screen must be generated at build time. As Aravis parses the GenICam XML from the camera in order to build its list of features, this XML is all that is needed to build a database and screen for the camera. Aravis provides a small utility that will allow the GenICam XML to be written to file.

The EPICS Database and EDM Screen

To parse the XML file, aravisGigE contains a Python script called makeDbAndEdl.py. The script creates a menu structure based on the Category nodes, and a list of feature nodes contained in it.

To create the EPICS database, for each feature node a suitable record is created, using stringin, longin, ai, and mbbi records for StringReg, Integer, Float, and Enumeration nodes respectively. If the field is editable the appropriate output record is created.

To create the EDM screen, a section is drawn for each category, then suitable widgets are created for the records that have been created. A tooltip is added with some suitable text, and a summary screen is created (see Fig. 4).

Figure 4: Section of Prosilica GC summary screen.

DISADVANTAGES OF REVERSE ENGINEERING

The main disadvantage of reverse engineering is that assumptions are made according to the cameras that are available. For instance, the author of Aravis only lists it as being tested on Prosilica and Basler cameras. It is no surprise then that the Prosilica cameras at Diamond work well with Aravis. Different vendors' cameras have been tested with varying degrees of success, the problems mainly being GenICam elements that are not supported in Aravis yet. There have however been some issues with dropped frames on some vendors' cameras that may point to a bug in the packet resend code.

CONCLUSION

The aravisGigE module is being rolled out on new beamlines, and GigE cameras have been retro-fitted on some existing beamlines. So far they have been more reliable than the Firewire cameras, and the simple cabling makes it much easier to move the cameras to different positions. Upcoming power over Ethernet versions of the cameras will reduce the number of cables still further.

It is very useful to have the ability to change manufacturer without the burden of having to write new software. Diamond's standard camera for beamlines is now the Allied Vision Technologies Manta, Fig. 5, but if a particular beamline has requirements that cannot be met by this range of cameras, then one can be sourced from another manufacturer. Aravis does an admirable job of controlling a range of cameras, given its reverse engineered origin.

Figure 5: Allied Vision Technologies Manta Camera.

REFERENCES

[1] Experimental Physics and Industrial Control System; http://www.aps.anl.gov/epics
[2] R. P. Walker, "Commissioning and Status of The Diamond Storage Ring", APAC 2007, Indore, India.
[3] Prosilica GC1020 Camera Specifications; http://www.alliedvisiontec.com/us/products/cameras/gigabit-ethernet/prosilica-gc/gc1020.html
[4] areaDetector: EPICS software for area detectors; http://cars9.uchicago.edu/software/epics/areaDetector.html
[5] ffmpegServer: video compression for areaDetector; http://controls.diamond.ac.uk/downloads/support/ffmpegServer
[6] AVT PvAPI SDK for GigE Vision® cameras; http://www.alliedvisiontec.com/us/products/software/windows/avt-pvapi-sdk.html

[7] GigE Vision® - True Plug and Play Connectivity; http://www.machinevisiononline.org/vision-standards-details.cfm?type=5
[8] The GenICam™ standard; http://www.emva.org/cms/index.php?idcat=27
[9] Aravis - A vision library for GenICam based cameras; http://live.gnome.org/Aravis
[10] EDM: an EPICS display manager; http://ics-web.sns.ornl.gov/edm
[11] aravisGigE: areaDetector GigE camera driver; http://controls.diamond.ac.uk/downloads/support/aravisGigE

ARCHITECTURE DESIGN OF THE APPLICATION SOFTWARE FOR THE LOW-LEVEL RF CONTROL SYSTEM OF THE FREE-ELECTRON LASER AT HAMBURG

Z. Geng#, SLAC, Menlo Park, California, U.S.A.
V. Ayvazyan, DESY, Hamburg, Germany
S. Simrock, ITER Organization, St. Paul lez Durance, France

Abstract

The superconducting linear accelerator of the Free-Electron Laser at Hamburg (FLASH) provides high performance electron beams to the lasing system to generate synchrotron radiation to various users. The Low-Level RF (LLRF) system is used to maintain the beam stabilities by stabilizing the RF field in the superconducting cavities with feedback and feed forward algorithms. The LLRF applications are sets of software to perform RF system model identification, control parameters optimization, exception detection and handling, so as to improve the precision, robustness and operability of the LLRF system. In order to implement the LLRF applications in the hardware with multiple distributed processors, an optimized architecture of the software is required for good understandability, maintainability and extendibility. This paper presents the design of the LLRF application software architecture based on the software engineering approach for FLASH.

INTRODUCTION

FLASH is a VUV and soft-X ray free-electron laser machine located at DESY, Hamburg. The layout of FLASH is depicted in Figure 1 [1].

FLASH requires high RF field stabilities up to 0.01 degree in phase (at 1.3 GHz) and 0.01% in amplitude [2]. But various perturbations, such as Lorenz force detuning, microphonics and thermal drift of the cable, will generate amplitude and phase errors on the RF field in the superconducting cavities of FLASH [3].

The LLRF system is introduced to maintain the RF field stabilities with both feedback and feed forward control schemes [4], see Figure 2. The feed forward control is used to control the repetitive perturbations like Lorenz force detuning, beam loading and slow drifts, while the feedback control is used to control the field errors caused by random perturbations like microphonics and charge variations along the bunch train.

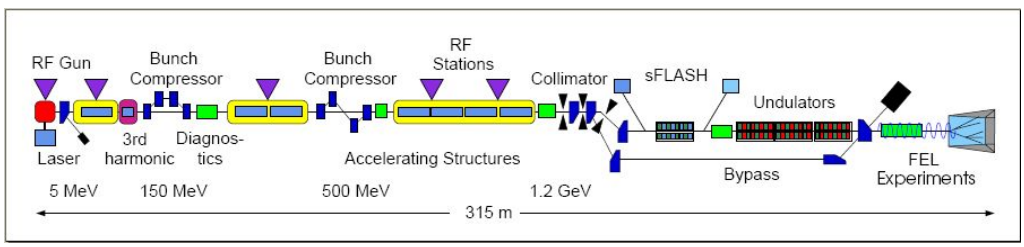


Figure 1: Layout of FLASH.

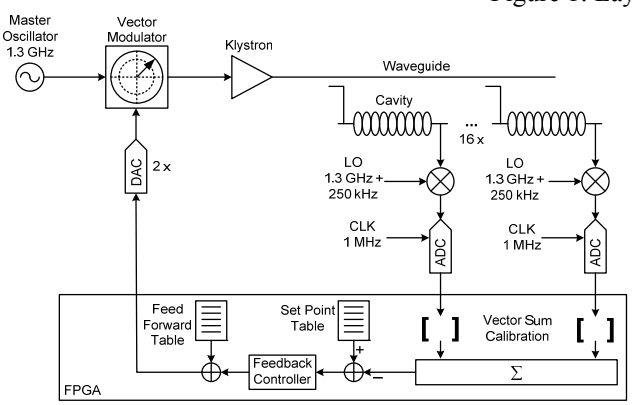


Figure 2: FLASH RF station including LLRF.

There are several great challenges for the LLRF system operation, including precise calibration of the vector sum for multi-cavity operation driven by a single klystron, optimal feedback and feed forward control, exception detection and handling for system robustness and automation for easy operation. So, sophisticated algorithms and application software are required to solve these problems in addition to the basic control loop.

LLRF APPLICATIONS

LLRF applications are sets of software to facilitate the LLRF control loop for better precision, robustness and availability. The main goals of the LLRF applications include

- Improve the RF field stabilities by optimizing the parameters of the RF field controller.
- Improve the robustness and availability of the LLRF system by system diagnostics, exception detection and handling.
- Support automation for easy operation.

The LLRF applications can be divided into several categories and their logical relations are show in Figure 3. Several important applications of each category are listed in Table 1.

#gengzq@slac.stanford.edu

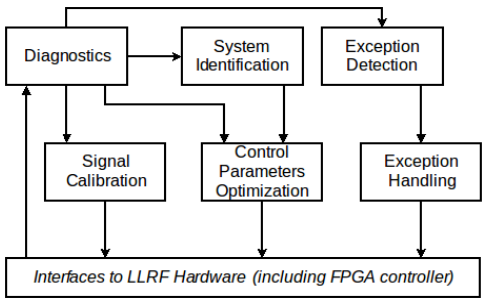


Figure 3: LLRF application categories.

Table 1: LLRF Applications

Category	Applications
Diagnostics	- Beam phase and beam current measurement - Loop phase and loop gain measurement - Cavity incident and reflected waves measurement - Cavity loaded quality factor and detuning measurement
Signal Calibration	- Vector sum calibration - Cavity gradient and phase calibration - Cavity incident and reflected power calibration - RF gun field calibration
System Identification	- RF system dynamic model identification - Cavity model identification - Klystron non-linearity characterization
Control Parameters Optimization	- Adaptive feed forward - Feedback gain scheduling
Exception Detection	- Cavity quench detection - Cavity operational limit exceeded detection - RF system components failure detection
Exception Handling	- Recovery from cavity quench - Adjust cavity gradient

SOFTWARE ARCHITECTURE DESIGN

Software Layers

The architecture of the LLRF applications is designed as several layers, see Figure 4.

The LLRF Algorithm Library will be realized in C language for better performance of the mathematic calculations. It contains most of the LLRF domain knowledge, and is optimized for computation power and memory consumption so that it can be highly reused for both DSP and common CPU programs.

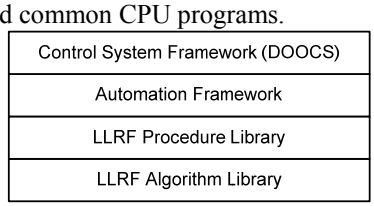


Figure 4: Layers of the LLRF applications architecture.

The LLRF Procedure Library will be realized in C++ language for better extendibility. It realizes the procedures for the tasks (use cases) of the LLRF applications. Sequential or state dependent logics will be used to coordinate the procedures. The platform independent parts of the procedures are designed as abstract classes, which are reusable for different machines with different platforms by derived classes, see Figure 5.

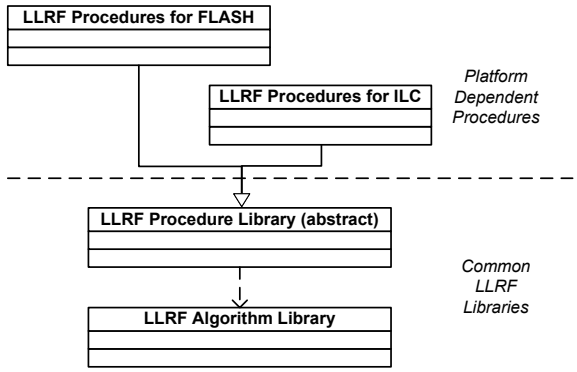


Figure 5: LLRF Libraries.

LLRF Algorithm Library

The aim of this LLRF algorithm library is to share the implementation with FLASH, European XFEL at DESY and even the future ILC.

To be maximum reused by various accelerators, the LLRF algorithm library is designed with C language, which can fit to the popular control systems like DOOCS [5], TINE [6], EPICS [7], and so on. And even can be used for DSP programming.

The library is broken down based on the domains, see Figure 6. The idea is to pack the code for a specified domain (like digital signal processing) so that the whole package can be reused in other applications. Of course, if one domain package has dependencies to other packages, the dependencies should be also considered for reusability.

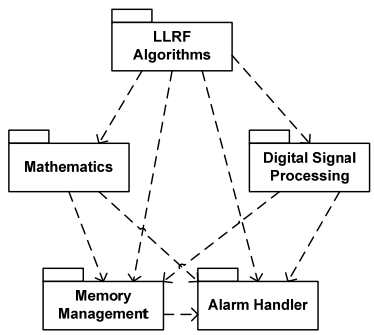


Figure 6: Domain breakdown of the LLRF Algorithm Library.

For each domain package, the codes are divided into three parts: “Available Interface”, “Library Body” and “Required Interface”. The interactions between domain packages are only through the interfaces.

The advantage of this design is that the library body can be upgraded without affecting other domain packages if the interfaces are not changed.

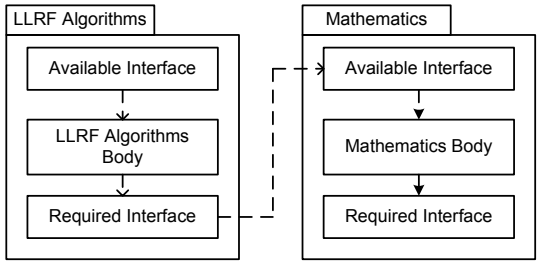


Figure 7: Dependency between packages.

Software Allocations

The LLRF applications at FLASH are realized in the DOOCS control system framework. Different applications are allocated to different DOOCS servers, and they use RPC to communicate with each other through the Ethernet.

The front-end servers are used for front-end hardware control, which will communicate with the LLRF hardware including FPGA controller. Some applications with real-time requirements are also allocated here to avoid the time consumption caused by Ethernet data transfer.

Most other applications for system calibration and optimization are realized in the middle-layer servers for each RF station. The applications that need to adjust more than one RF station are allocated to the middle-layer server for global control. One example is the recovery from quench, from which the cavity gradient of more than one RF stations may be changed.

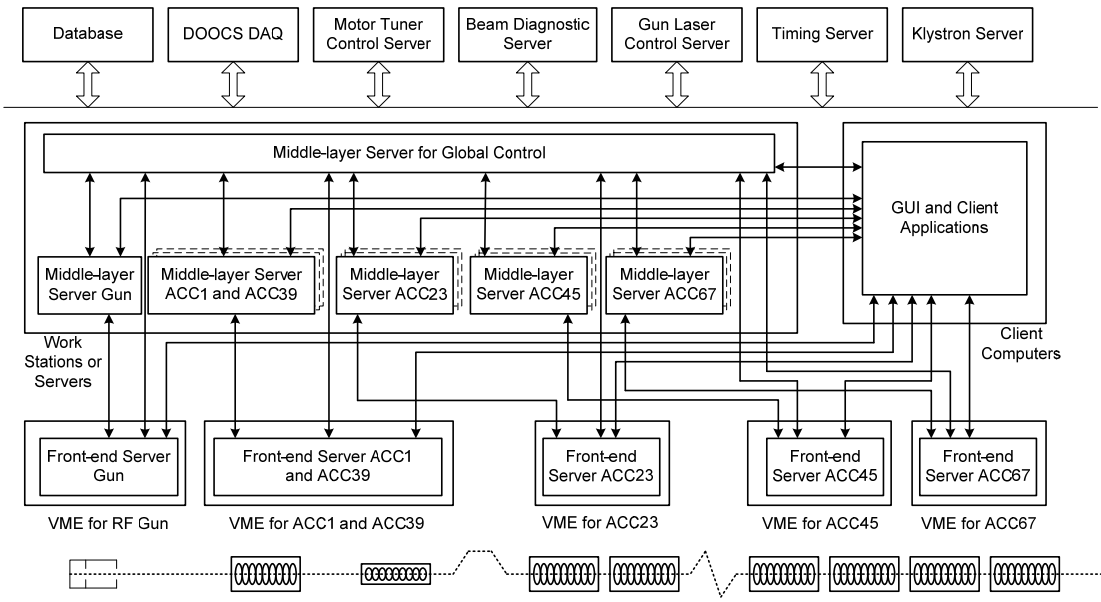


Figure 8: Allocation of the LLRF applications.

CONCLUSION

Software architecture is designed for LLRF applications at FLASH. This is the first time to implement the LLRF applications in a systematic way. The LLRF Algorithm Library has been developed and several applications have been successfully implemented and tested at FLASH [8], including the control table generation, vector sum calibration, RF gun calibration, RF system identification and loop phase and loop gain control. The architecture is proved successful for good understandability, maintainability and extendibility, which will be the reference design for the application software of the LLRF system for the European XFEL project. The experiences gained are also useful for the LLRF system design for other machines like ILC and normal conducting accelerators.

REFERENCES

[1] http://flash.desy.de/
[2] M. Hoffmann. Development of a multichannel RF field detector for the Low-Level RF control of the Free-Electron Laser at Hamburg. PhD thesis. 2008
[3] S. Simrock, Z. Geng. Sources of Field Perturbations, LLRF lecture for the Forth International Accelerator School for Linear Colliders.
[4] S. Simrock, Z. Geng. Cavity Field Control, LLRF lecture for the Forth International Accelerator School for Linear Colliders.
[5] http://tesla.desy.de/doocs/doocs.html
[6] P. K. Bartkiewicz et al. The TINE Control System, Overview and Status, ICALEPCS'07, Knoxville, Tennessee, USA, 2007
[7] http://www.aps.anl.gov/epics/
[8] V. Ayvazyan, K. Czuba, Z. Geng et al., "Low Level RF Control System Upgrade at FLASH", in Proceedings of PCaPAC 2010, Saskatoon, Canada, 2010

USE OF ITER CODAC CORE SYSTEM IN SPIDER ION SOURCE*

A. Barbalace, M. Breda, R Capobianco, A. Luchetta, G. Manduchi, F. Molon, M. Moressa, P. Simionato, C. Taliercio#, Consorzio RFX, Associazione Euratom-ENEA sulla fusione, Corso Stati Uniti, 4 - 35127 Padova - Italy

Abstract

In February 2011 ITER released a new version (v2) of the CODAC Core System. The SPIDER Ion Source experiment is the first experiment planned in the ITER Neutral Beam Test Facility under construction at Consorzio RFX, Padova, Italy. As the final product of the Test Facility is the ITER Neutral Beam Injector, we decided to adhere from the beginning to the ITER CODAC guidelines. Therefore the EPICS system provided in the CODAC Core System will be used in SPIDER for plant control and supervision and, to some extent, for data acquisition. In this paper we report our experience in the usage of CODAC Core System v2 in the implementation of the control and data acquisition (CODAS) system of SPIDER and, in particular, we analyze the benefits and drawbacks of the Self Description Data (SDD) tools.

INTRODUCTION

The ITER plasma will be heated up to ignition by dedicated additional heating and current drive systems based on the injection of radio frequency power (ion cyclotron, electron cyclotron, lower hybrid) and neutral beams in H or D, which are produced by accelerating negative ions that are successively neutralized. As the performance requirements for the ITER Heating Neutral Beam Injectors (HNB) (beam power 16MW, energy up to 1MeV, beam-on time up to 3600s) are far beyond the parameters of existing injectors [1,2,3], ITER will support the development of the ITER HNBs by an ad-hoc ITER Neutral Beam test facility, to be constructed in Padova, Italy. The first test bed of the facility, referred to as SPIDER, will be used for the development of the ITER full-size ion source.

As the final product of the Test Facility is the ITER Neutral Beam Injector, we decided to adhere from the beginning to the ITER CODAC guidelines. Therefore the EPICS system provided in the CODAC Core System will be used in SPIDER for plant control and supervision and, to some extent, for data acquisition. ITER currently does not provide guidelines neither for the management and the storage of experiment data, nor for the implementation of fast control systems. For this reason, two other frameworks will be integrated in SPIDER: MDSplus and MARTe. MDSplus [4] is a framework for Data Acquisition and management largely used in the fusion community. MARTe [5] is a framework for fast real-time control that has been used in several control applications in JET and other fusion machines in Europe. Interfaces between EPICS, MDSplus and MARTe have been developed to provide seamless integration in a complete Data Acquisition and Control system.

An important concept in ITER CODAC is the Self Description Data (SDD), i.e. is the formal description of all the information related to CODAC. Such information includes the definition of the data items, the used hardware, and the software configuration. XML has been chosen to express self-description data, as it is application-independent and is purely data-centred. Even if in principle all self-description data can be expressed by XML text files, in practice more efficient storage strategies are chosen. In particular, the PostgreSQL relational database has been used in ITER to store SDD information. The current version of CODAC Core System includes also an SDD editor, which provides a user friendly interface for the definition of the data items in SDD. Not all aspects of Self Description are currently implemented because the technology of several CODAC components has not yet been chosen by ITER. Currently the SDD editor allows the definition of the functional breakdown, the signal list, and the variables and commands in controllers.

The information stored in the SDD database is used to generate the EPICS configuration files and the PLC Data blocks for data and command exchange between EPICS Input/Output Controllers (IOCs) and the PLCs. It is also possible to export the SDD in XML, and this feature will be used in SPIDER to add configuration generation tools for MDSplus, with the aim of making SDD the sole repository for the entire system configuration.

In the rest of the paper presents the software frameworks used in SPIDER, in particular those components that have been developed to achieve their integration. The applicability of the SDD approach in SPIDER CODAS will be then discussed, as well as the initial experience in the usage of the SDD editor and, more in general, the implications of the Self Description in the integration of different systems.

USED FRAMEWORKS AND THEIR INTEGRATION

Three open-source software frameworks will be used for the implementation of SPIDER Central CODAS and Plant System CODAS: EPICS, MDSplus and MARTe. The choice of three different frameworks is a consequence of the fact that no framework fully covers the requirements for SPIDER CODAS. EPICS addresses overall coordination and slow control, but does not provide a flexible management for data acquisition, nor is it designed for efficient, fast real-time control.

*This work was set up with financial support by Fusion for Energy.
#cesare.taliercio@igi.cnr.it

Sophisticated data management and storage are instead provided by MDSplus, while MARTe provides the functionality required for the implementation of hard real-time control systems.

EPICS

EPICS is a software framework developed to control particle accelerators and other physics experiments. The core of EPICS consists basically of two components: Process Variables (PV) management and Input/Output Controllers (IOC). Control and monitoring in EPICS is achieved by IOCs, where every IOC runs as an application and runs the operations specified by a set of records defined in one or more EPICS databases (EPICS DB). An EPICS DB defines the required actions for control and monitoring. A set of record types is available and can be used for operations such as reading and writing PLC data blocks. Several other tools are part of the EPICS package, including interactive generation of graphical interfaces, alarm handling and data archiving. These tools are built over Eclipse and form the Control System Studio [6] (CSS), available in the ITER MiniCODAC distribution

MDSplus

MDSplus is a software framework for the management of data and the supervision of data acquisition in fusion experiments. MDSplus is centred on the concept of *pulse file* that is a database containing all information which is pertinent to the experiment, including configuration data and experimental results. A large variety of data types is supported by MDSplus including the Signal data type, which describes the time evolution of signals, normally acquired by Analogue to Digital Converters (ADCs). In addition to data describing the experiment parameters and the experimental results, MDSplus provides the Device data abstraction. A Device in MDSplus is represented by a set of data items describing the current configuration of a given hardware component. Multiple instances of the same kind of hardware device will be reflected in the pulse file by multiple instances of data sets, with the same structure, each describing the current configuration of the associated device instance. A set of methods is associated with every device type, corresponding to a routine reading or writing device data items and interacting with the underlying hardware. A Dispatcher tool supervises the execution of all the required operations in the right order during the experiment sequence.

The foreseen usage of MDSplus in SPIDER CODAS is the management of all the configuration and experimental data, including data storage, data access, graphical visualization of acquired signals, and the supervision of the acquisition of data not coming from PLC. Whilst interaction with PLCs in SPIDER will be by means of EPICS IOCs, the supervision of the other hardware devices, such as ADCs and Camera devices will be handled by MDSplus devices, orchestrated by a set of dispatchers, each supervising the operation of a given control unit.

EPICS-MDSplus Integration

As all data dealt with by SPIDER CODAS resides in MDSplus, it is necessary to export configuration and setup parameters to EPICS. This is achieved by letting MDSplus data be exported as PVs via a Channel Access (CA) server. In SPIDER CODAS all the configuration data items held in the experiment model are exported as PV and therefore they are natively available whenever they are required (e.g. to send configuration data to a PLC via the corresponding PLC interface record in the IOC supervising the associated unit).

Exporting configuration data as PVs allows also the interactive development of graphical interfaces for operator displays, carried out by BOY, a component of the EPICS CSS toolset.

Data which are produced in EPICS IOCs will be stored in MDSplus pulse files, too. Such data will be stored in a pulse file by the EPICS – MDSplus Channel Archiver [6], a tool similar in functionality to the BEAUTY tool of the EPICS CSS. BEAUTY provides the monitoring of a selected set of PVs, storing PV values, acquired either at regular rates or whenever their values change, in a relational Database. The MDSplus Channel Archiver has the same interface towards the selected PVs (both tools are CA clients) but stores PV values in MDSplus pulse files. A PV is associated with every EPICS record, and therefore data values acquired from a given PLC, supervised by the corresponding EPICS records, are mapped onto the associated PV variable, and then automatically changed in the trend pulse file.

As EPICS is used in SPIDER CODAS for the overall coordination during the beam sequence, it is necessary to "trigger" the execution of actions supervised by MDSplus at the right time during the beam sequence. As stated before, the specification of the actions to be executed is embedded in the pulse file. The MDSplus Dispatcher will collect such information and will dispatch the corresponding actions whenever requested to execute a given phase in the beam sequence. Such a request must come from an EPICS IOC running the state machine for the SPIDER beam sequence, and will be sent to MDSplus via a dedicated EPICS record.

MARTe

MARTe is a framework for real-time control. MARTe orchestrates the execution of real-time threads, possibly running on different cores of a multi-core system, and the associated data flow. Every thread carries out the execution of a set of Generic Application Modules (GAMs), handling data acquisition from sensors (normally ADC devices), the computation of some control algorithm, and the generation of the reference signals for the plant. MARTe provides an abstraction of the underlying operating systems and therefore can be run on different platforms. New GAMs can be easily integrated in the system, provided they adhere to a given interface, and the whole system is dynamically configured

by a configuration file, thus providing rapid integration of new control components in the system.

The functionality of MARTe is partially overlapping with EPICS IOCs, as real-time control configuration can defined as one or more EPICS DBs and run by one IOC. However the footprint of MARTe is much more reduced compared with that of EPICS, and MARTe can handle fast control with a number of inputs/outputs well beyond the limits of EPICS [8].

The foreseen usage of MARTe in SPIDER CODAS is the generation of the power supply reference signals, including the management of breakdowns, requiring the interruption of the voltage generation and its subsequent restart with a given dynamics whenever a breakdown is detected.

MARTe integration

The required functionality for real-time control in SPIDER CODAS (initially limited to the generation of the power supply reference signals and the management of breakdowns) will be carried out by a set of GAMs developed for this purpose. These GAMs will acquire in real-time one or more signals carrying information about breakdowns and will produce the reference signals for the power supplies, which have to be turned off and then gradually on in real-time. In addition to the Input/Output signals these GAMs will likely require some configuration parameters and will write some signals (e.g. describing the internal behaviour of the control algorithm) in the pulse file. The required data exchange will be achieved via MDSplus as all data dealt with by the experiment is hosted in MDSplus pulse files

INTEGRATION OF CODAC CORE SYSTEM

The current version (v2) of CODAC Core System basically consists in the EPICS core and CSS components, and the SDD editor.

The SDD editor provides a graphical interface for the definition of the Self Description database, forcing at the same time a set of rules for naming convention in order to adhere to the ITER standards. The current version of the SDD editor covers only a subset of the Self Description Data, represented by those components which are required for the generation of the configuration files for EPICS and the data blocks for the declared PLCs.

The SDD provides Schemas for Plant Systems and Plant Systems I&C. A plant system is an autonomous part of the ITER plant which implements a given technical function. Self Description for Plant Systems will define those physical components which are relevant for Instrumentation and Control (I&C). The list of signals handled by the system is part of this description. Every signal, characterized by a name adhering to the ITER naming conventions, is described by properties describing the component to which it refers, its physical characteristics, such as the physical units, and its I&C characteristics, such as its digital resolution.

Whilst the Plant System description provides a description of the physical organization of the described Plant, a different view is normally required when considering the logical organization of its I&C. Such organization is captured in the Plant System I&C part of the SDD schema. Plant System I&C description includes the I&C components, which may be represented by:

- Slow and fast controllers;
- I/O boards;
- PLCs

In addition, Plant System I&C Self Description includes all the EPICS Process Variables (PVs) which are involved in control and data acquisition.

In order to ease the development of SDD, the SDD editor provides a simplified interface which hides the internal distinction between Plant System and Plant System I&C. Basically, for every plant system, the SDD editor creates three folders:

- *Signals*: defined by a set of components, each containing zero or more signals. Components describe physical parts of the plant system. Signal definitions here refer to the physical description of the input or output systems handled by the component. All information defined under the Signal folder is saved in the Plant System part.
- *Variables & Commands*: describing the Process Variables used to exchange data with the PLCs and the I/O boards as well as the commands to be issued to the controllers. Variables and Commands are organized in functions, which describe the functional breakdown of the Plant I&C. This information is stored in the Plant System I&C part of Self Description.
- *Control Units*: describing the control components (Fast Controllers, I/O modules and PLCs) involved in the Plant System I&C.

An extensive consistency check is carried out by the SDD editor to ensure the consistency of the entered information, including the adherence to the ITER naming convention and the cross references. For example, a Process Variable (defined in Variables & Commands) must be deployed to an existing target (defined in Control Units), and a signal must be associated either with an I/O module channel or with a PLC.

Once the I&C configuration has been entered and saved in PostgreSQL, the required configuration files can be generated. Currently, the EPICS IOC Database description is generated, which defines the EPICS records corresponding to the declared Process Variables, and the CSS configuration files for the alarm interface (BEAST), the Operator Panel (BOY) and the channel archiver (BEAUTY). In addition the data block files for the PLC configuration are generated.

SDD USAGE IN SPIDER

The ITER approach in the definition of the plant I&C via the SDD has proved extremely useful in our

experience, and the availability of a good SDD editor from the very beginning of the system development represents a key factor. Being forced to declare all I&C related information using a single tool simplifies the development process, preventing the scattering of information among different components, and avoids to a large extent errors related to the inconsistent mapping among subsystems. It forces at the same time a strict usage of the naming rules, a very important factor in the development of a large system. Moreover, such an approach allows the automatic generation of a set of useful support tools for system monitoring. As an example, besides the generation of the IOCs based on the current definition of the Process Variables and I/O channels, the CODAC Core system generates also two IOCs for the monitoring of several system parameters such as processor load and the state of the communication link with the PLCs.

We found however two main limits in the current tools for SDD management. The first one is that this approach tends to produce a "flat" model, where a set of records is generated, mapped to the I/O channels (PLCs of other boards) defined in the system. Additional semantics, such as the implementation of specific control in the generated IOCs, or the implementation of synchronizing State Machines can currently be achieved only by manually configuring the EPICS DBs to describe the additional functionality. The drawback of such an approach is that whenever SDD information is changed (e.g. a new signal is added), and the new configuration regenerated, all the additional user-provided configuration information is lost. This is a general problem in the automated generation of software components, which could be solved only by incorporating in the SDD definition also the control semantic. This is however a very ambitious approach because it is very hard to foresee all the possible control approaches, and in practice may lead to a system that is too rigid to be usable in practice.

The second limit is due to the fact that the current version is entirely EPICS centred. While the usage of EPICS has a large consensus for plant supervision and monitoring and, more in general to handle plant (slow) signals, the usage of EPICS for fast control and data acquisition may lead to inefficient solutions. Consider for example the acquisition and the storage in the experiment database of a set of signal acquired by an ADC board with a number of analogue inputs. When supervising data acquisition in EPICS, an Analog Input (ai) record would be defined for the ADC board, whose associated driver handles communication with the ADC hardware. The corresponding PV variable would then be monitored by a BOY Channel Archiver tool, saving every sample in a database. Whenever a new ADC sample is available, the corresponding record will be activated, and the value of the associated PV exported via the Channel Access (CA) protocol over the network and finally received and stored in the experiment database. If the sampling rate is of several kHz and many signals are involved, the system overhead may soon become unsustainable. The approach taken in MDSplus, the Data Acquisition system chosen for SPIDER, is completely different, and a data acquisition thread is created to supervise the acquisition and storage of all the channels of the ADC. Whenever a new set of data samples is available, the thread will directly store the data samples in the pulse file using optimized disk access and network communication, (in the case data access is remote). MDSplus has been in fact designed to handle very high data throughput and is routinely used in fusion experiment where data rates of hundreds of MB/s are achieved.

The SDD approach turns out to be useful even in the implementation of the non-EPICS parts of SPIDER CODAS, and the XML representation of SDD will be used as the source of the structure of the MDSplus pulse files. More in detail, information like the list of signals and the definition of the I/O modules can be retrieved and used to produce a standardized structure of the pulse files. This is achieved by using EXtensible Stylesheet Language [9] (XSL) to define the transformation of the Self Description XML files into the XML files used in MDSplus for the definition of the pulse file structure.

REFERENCES

[1] Y. Ikeda et al., "Present status of the negative ion based NBI system for long pulse operation on JT-60U", Nucl. Fusion 46 No 6 (2006) S211-S219.

[2] K. Ikeda et al., "Recent Progress of Neutral Beam Injector and Beam Emission Diagnosis in LHD", Plasma Sci. Technol. 11 No 4 (2009) 452-455.

[3] A. Staebler et al., "Development of a RF-driven ion source for the ITER NBI system", Fus. Eng. Des., 84 (2009) 265-268.

[4] MDSplus, web page http://www.MDSPlus.org/

[5] A.C. Neto, et al, MARTe: A Multiplatform Real-Time Framework, IEEE Transactions on Nuclear Science, No 57 (2010)479-486.

[6] CSS Home Page at http://css.desy.de/content/index_eng.html

[7] G. Manduchi, A. Luchetta, C. Taliercio, A. Soppelsa, A. Barbalace "New EPICS Channel Archiver based on MDSplus Data System" to appear in IEEE Transactions on Nuclear Science

[8] A. Barbalace, G. Manduchi, A. Neto, G. De Tommasi, F. Sartori, D. F. Valcarcel "Performance Comparison of EPICS IOC and MARTe in a Hard Real-Time Control Application" to appear in IEEE Transactions on Nuclear Science

[9] XSL Home page at http://www.w3.org/Style/XSL

INTUITIONISTIC FUZZY (IF) EVALUATIONS OF MULTIDIMENSIONAL MODEL

Ivanka Valova
Sofia University, Sofia, Bulgaria

Abstract

There are different logical methods for data structuring, but no one is perfect enough. Multidimensional model-MD of data is presentation of data in a form of cube (referred also as infocube or hypercube) with data or in form of "star" type scheme (referred as multidimensional scheme), by use of F-structures (Facts) and set of D-structures (Dimensions), based on the notion of hierarchy of D-structures. The data, being subject of analysis in a specific multidimensional model is located in a Cartesian space, being restricted by D-structures. In fact, the data is either dispersed or „concentrated", therefore the data cells are not distributed evenly within the respective space. The moment of occurrence of any event is difficult to be predicted and the data is concentrated as per time periods, location of performed business event, etc. To process such dispersed or concentrated data, various technical strategies are needed. The basic methods for presentation of such data should be selected. The approaches of data processing and respective calculations are connected with different options for data representation. The use of intuitionistic fuzzy evaluations [1]- IFE provide us new possibilities for alternative presentation and processing of data, subject of analysis in any OLAP application. The use of IFE at the evaluation of multidimensional models will result in the following advantages: analysts will dispose with more complete information for processing and analysis of respective data; benefit for the managers is that the final decisions will be more effective ones; enabling design of more functional multidimensional schemes.

The purpose of this work is to apply intuitionistic fuzzy evaluations of multidimensional model of data.

Key words: On-line Analytic Processing, Intuitionistic Fuzzy Evaluations.

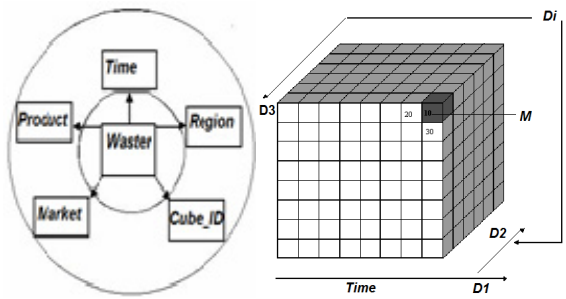
INTRODUCTION

The main task at development of MD model referred as OLAP models is, on the one hand to make the data schemas more understandable for end users and on the other hand to improve the performance in case of queries placed by such users. For such purposes the data schemes are simplified in a manner to contain only most important things (i.e. fact that is to analyzed and respective D-structures, which will be involved in data analysis). These schemas are very close to the concept for performance of data analysis and require the use of specific types of queries, enabling easy adjustment of the system to the needs of users, aiming the obtaining of better response times and producing of result in case of specific query placed.

DESCRIPTION OF THE MODEL

Three different levels of detail have been distinguished in the establishment of a data model, intended for OLAP. On a low level there are measures, which can be grouped into cells if they refer to the same fact (F-structure). The cells further are grouped into different classes, which can be drawn as n-dimensional cubes (on a medium level of detail), owing to which the different dimensions, defining the cube are functionally independent. Eventually, on a high level several cubes representing the same type of fact on different levels of aggregation are grouped into one Fact.

The OLAP model must be able to present a global view of data, including full support for hierarchies of D-structures and multiple hierarchies of data, subject to analysis.



1a) 1b)

Figure 1: Example of OLAP model. 1a) Multidimensional scheme-one F-structure (fact) and several Dimensions (D-structures), 1b) Infocube.

On Fig. 1 is specified example of an OLAP model, where the fact is Wasted product and D-structures are Time, Info Package, Cube_ID, Region, etc. In Fig.1 the Fact is shown in the middle of the scheme, and D-structures (dimensions), being used for analysis of the Fact, are shown around it. The scheme presented in Fig. 1 is also termed "multidimensional scheme" due to the large number of D-structures, which may be involved. The fact in a multidimensional scheme is the object, which contains measures.

Organization of data in a "star" type schema or its representation in form of a cube is very close to the concepts for performance of analysis and facilitates the use of data by end users.

Description of the D-structures

In multidimensional models the information is divided in facts (F-structures) and dimensions (D-structures) [2].

D-structure, (Dimension) – different initial viewpoints at data selection, which will be used during fact analyzing. D-structures contain mainly description attributes.

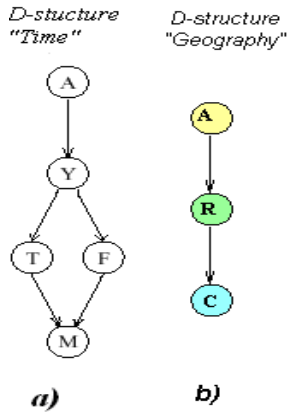


Figure 2: a) D-structure "Time" , A-All, Y- Year, T-three month, F-four month, M –Month b) D- structure "Geography", A-All, R- Region, C-City.

D-structure represents a connected directed graph, provided that each node of the graph corresponds to a given aggregation level, and its arcs reflect "part-whole" relations between the objects within the aggregation levels.

The hierarchy of certain D-structure is linked with determined data consolidation path (Fig. 2). The consolidation path is composed of series of levels or consolidation steps, where each level represents a higher degree of data consolidation. For instance, "business enterprise" may include as consolidation path the levels "business area", "branch", "department", "project", "task" and "employee" [3]. The software product SAP BIW supports 16 D-structures, provided that three of it are mandatory. These are: "Time", "Units", which provide meaning of the values for key indicators and Info packages.

Description of the F-structures

F-structure (Fact) – it represents the data, subject to analysis. The fact contains numerical attributes. The fact in a multidimensional scheme is the object, which contains measures. Measures evaluate attributes of fact [4].

The fact wasted products is evaluated by using the following measures: costs for invested labor, costs for materials used, transportation costs, number of persons involved in the process of wasting and evaluation of the respective products and total costs and other depending on the point of view of the developer and on the endmost needs of analysts and users.

We have evaluated D-structure Product (the degree of preference for whether an article is liked or not), and to make our evaluation more complete it is necessary to consider the Region, where these goods are wasted, the Chain of stores in which the largest quantities have been wasted (and to try to answer whether the wasted goods are a result from poor management), the Vendor who has wasted the goods (if wasted goods are in found in greatest quantity in the same vendor) and if we also consider the Time as a D-structure, we can evaluate in which periods the quantity of wasted goods is the greatest and thus we can evaluate the degree of consumption depending on different seasons.

The subject matter of analysis (fact) is wasted products. Typical D-structures of this application are: product, which is wasted, the store in which it is wasted (e.g. Sofia, Z street) and the date of the respective operation. These D-structures form one hypercube.

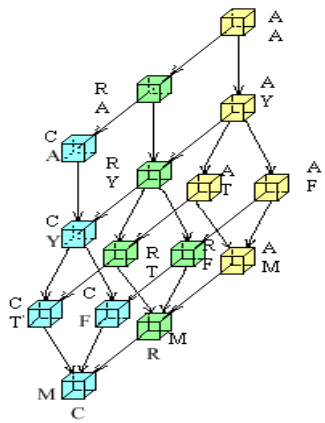


Figure 3: Example of F-structure.

In order to simplify the drawing of model only two D-structures are used – Time Fig. 1a), Geography –Fig. 1b). It is shown on Fig. 3 the structure of Fact, composed of several cubes (boxes). The cubes containing letter C may be used for the purposes of analysis of goods in certain city for different time periods – month, quarter..... year. The cubes containing letter R are used for analysis of goods per regions, again for month, quarter..... year. The box MC shows goods for a month in given city. The cube CT shows certain goods in given city and for time period - quarter. The cube MR reflects the certain goods for a month, but in given region, etc.

If in any F- structure, intended for OLAP, exist All-level and atomic level and the remaining levels are segmented (i.e. contain more than one object or more than one cube), than the graph is a lattice. In the F-structure, intended for OLAP, exists unique level called atomic, which contains elementary objects. Each object at atomic level has one part and this object is called elementary. In D – and F-structures, intended for OLAP, exists a unique level called All- level. This aggregation level has exactly one object.

INTRODUCTION OF IF ASSESSMENTS

We use A_1, A_2,..,.A_n for marking of articles (attributes of certain product). Each symbol may be indicated by lower index for marking of different names from same type.

Each attribute A should be related with definition area dom(A).
Let:
$k(A_i, R_i, T_i)$ – to be the quantity of the objects of the relevant product.
$p(A_i, R_i, T_i)$ – the quantity of the sold articles
$q(A_i, R_i, T_i)$- the quantity of the waste articles.
T_i, R_i – specify time and location of the relevant article

$$\{ \langle A_i, R_i, T_i \rangle, \mu(A_i, R_i, T_i), \nu(A_i, R_i, T_i) \mid \langle A_i, R_i, T_i \rangle \in E \}$$

π - the produced articles, but still remaining unsold, i.e. some of them may be scrapped, therefore there is indefiniteness.
μ - produced and sold articles
V_ν *(All)* - gives the correlation of the waste articles towards the overall production.

$$V_\nu(All) = \frac{\sum_{i=1}^{n} q(A_i, R_i, T_i)}{\sum_{i=1}^{n} k(A_i, R_i, T_i)} \quad (1)$$

$V_\nu(A_i, R_i, T_i)$ - assigns the degree of the waste articles.

$$V_\nu(A_i, R_i, T_i) = \frac{q(A_i, R_i, T_i)}{k(A_i, R_i, T_i)} \quad (2)$$

$V_\mu(All)$ - give preference to the relevant article as a whole

$$V_\mu(All) = \frac{\sum_{i=1}^{n} p(A_i, R_i, T_i)}{\sum_{i=1}^{n} q(A_i, R_i, T_i)} \quad (3)$$

$V_\mu(A_i, r_i, t_i)$ – assigns the degree of preference, whether certain article is preferred or not.

$$V_\mu(A_i, R_i, T_i) = \frac{p(A_i, R_i, T_i)}{k(A_i, R_i, T_i)} \quad (4)$$

From (1) and (3) it follows that

$$V_\mu(All) + V_\nu(All) = \frac{\sum_{i=1}^{n} p(A_i, R_i, T_i) + q(A_i, R_i, T_i)}{\sum_{i=1}^{n} k(A_i, R_i, T_i)} \le \frac{\sum_{i=1}^{n} k(A_i, R_i, T_i)}{\sum_{i=1}^{n} k(A_i, R_i, T_i)} = 1 \quad (5)$$

Therefore $\langle V_\mu(All), V_\nu(All) \rangle$ is an IF estimation.

From (2) and (4) it follows that

$$V_\nu(A_i, R_i, T_i) + V_\mu(A_i, R_i, T_i) = \frac{q(A_i, R_i, T_i) + p(A_i, R_i, T_i)}{k(A_i, R_i, T_i)} \le \frac{k(A_i, R_i, T_i)}{k(A_i, R_i, T_i)} = 1 \quad (6)$$

for each A_i, R_i, T_i.
Therefore $\langle V_\mu(A_i, R_i, T_i), V_\nu(A_i, R_i, T_i) \rangle$ is an IF estimation.

For evaluation of the model we can use Fig. 4. We introduce β, α - Fig. 4. If the aggregate evaluation falls within the trapezoid αODB we have the optimal variant. In case that the respective evaluation falls within the trapezoid αECA, it serves as a sign that the production is ineffective. If the aggregate evaluation falls within the triangle EOD, it means that the production output exceeds the minimum and is sold above the minimum, and on other side the goods become slow-moving ones, which serves as indicator that it is produced more than needed.

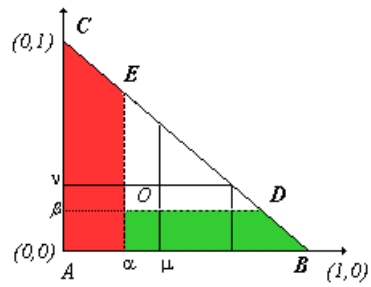


Figure 4: Intuitionistic fuzzy evaluations.

The use of IFE at the evaluation of multidimensional models will result in the following advantages: analysts will dispose with more complete information for processing and analysis of respective data; benefit for the managers is that the final decisions will be more effective ones; enabling design of more functional multidimensional schemes.

CONCLUSIONS

In this paper we represented the use of IFL in assessment of the MD models for analytical data processing.

REFERENCES

[1] Atanassov K., Intuitionistic Fuzzy Sets, Springer, Heidelberg, 1999.
[2] Valova I., M. Noirhomme-Fraiture, Processing of Large Data Sets: Evolution, Opportunities and Challenges, PCaPAC'08, Ljubljana, Slovenia, 2008, 198-200.
[3] Codd E., S. B. Codd, C. T. Salley. Providing OLAP to user-analysts: An IT mandate. Technical report, E. F. Codd & Associates, 1993.
[4] Weidner Jens -Integrierte Data Mining Lösung mit SAP BI, (SAP BW3.5 / SAP CRM4.0), CRM Marketing & Analytics SAP.
[5] Kimball R., Data Warehouse Tool Kits. John Wiley & Sons, Toronto, 1996.

NOMAD – MORE THAN A SIMPLE SEQUENCER

P. Mutti, F. Cecillon, A. Elaazzouzi, Y. Le Goc, J. Locatelli, H. Ortiz, J. Ratel,
Institut Laue-Langevin, Grenoble, France

Abstract

NOMAD is the new instrument control software of the Institut Laue-Langevin (ILL). A highly sharable code among all the instruments' suite, a user oriented design for tailored functionality and the improvement of the instrument team's autonomy thanks to a uniform and ergonomic user interface are the essential elements guiding the software development. NOMAD implements a client/server approach. The server is the core business containing all the instrument methods and the hardware drivers, while the GUI provides all the necessary functionalities for the interaction between user and hardware. All instruments share the same executable while a set of XML configuration files adapts hardware needs and instrument methods to the specific experimental setup. Thanks to a complete graphical representation of experimental sequences, NOMAD provides an overview of past, present and future operations. Users have the freedom to build their own specific workflows using intuitive drag-and-drop technique. A complete drivers' database to connect and control all possible instrument components has been created, simplifying the inclusion of a new piece of equipment for an experiment. A web application makes available outside the ILL all the relevant information on the status of the experiment. A set of scientific methods facilitates the interaction between users and hardware giving access to instrument control and to complex operations within just one click on the interface.

INTRODUCTION

The Instrument Control Service (SCI) in close collaboration with the ILL's scientists has taken up the challenge to redefine the way experiments are performed. The initial part of the project has been dedicated to an exciting discussion with our scientists and users to define all the use-cases to be included in NOMAD. Such a discussion brought the attention on the following use-case:

UC1: *hardware installation*. Hardware added to the instrument needs to be incorporated into the software control. We have detailed how a new electronic device can be added dynamically to the control software.

UC2: *instrument component setup*. Once a new hardware is added it needs to be configured. This use-case takes into account the different roles that hardware components can have on the instrument control.

UC3: *instrument calibration*. After the hardware is correctly installed and configured, it needs to be calibrated to ensure a correct functioning of the instrument before performing an experiment. This may involve proper PID settings, detector calibration, etc…

UC4: *sample alignment*. In certain cases, a sample needs to be aligned. This is most commonly performed on diffraction-type instruments in which the diffraction pattern depends upon the orientation of the sample with respect to the neutron beam. In other types of instruments this alignment is sometimes known in advance but the instrument still needs to be setup correctly depending on the type of experiment being performed.

UC5: *performing the experiment*. This use-case describes the general steps involved in the execution of an experiment. It starts from the verification of the status of the instrument to continue with the execution of pre-defined list of commands. At the end of a well-defined sequence, data are stored and a detailed report of all the various steps undergone is produced.

All those different use-cases lead to the implementation presented in the current paper.

ARCHITECTURE

The Nomad application is a two-tier client server application [1]. Figure 1 shows a simplified component diagram of the application

Server

The Nomad Server application is the part of the program executing the main tasks required for the control of the instruments. The server written in C++ [2] has three main roles:

- Data provider: to provide the actual values of the variables of the system (e.g. the properties of the instrument).
- Plug-in container: to load, initialize and execute the business logic of the instrument. The state of the instrument is periodically updated by the reading of the connected devices. The business layer is organized into plug-ins separated into drivers (low level layer) and controllers (high-level layer).
- Command sequencer: to process the requests of the client. Those can be complex sequences of instructions onto controllers. A client user can launch batch processing by programming a combination of for loops with parallel execution of commands for the night.

The plug-in libraries are divided into controllers and drivers. These are dynamic libraries loaded by the server. They contain all the business code written for NOMAD including all the available drivers and controllers. Some controllers can be common to each instrument, but some controllers are instrument specific. We also have different levels of abstraction for controllers.

The controller and driver classes loaded by the plug-in libraries describe an object-oriented model of classes. Controller classes interact with controller or driver

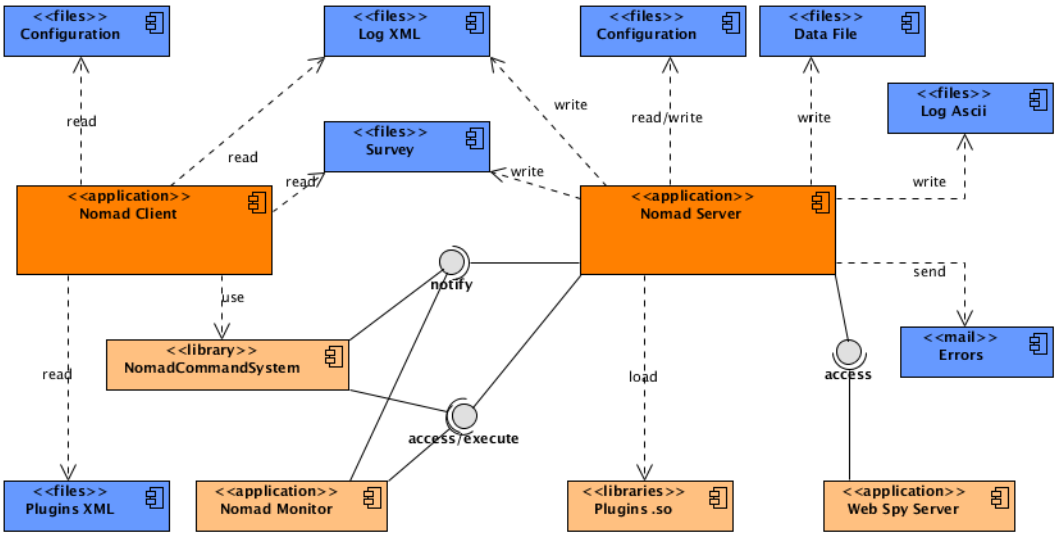


Figure 1: Simplified component diagram for the NOMAD application.

interfaces. We need to describe how we link the different controller and driver objects. These configurations are specific to each instrument. Two main configuration files, the *InstrumentConfig* and *HardwareConfig*, list the controllers and drivers required for a given instrument and how they are linked together. Additional instrument specific configuration files describes acquisition settings or scheduler rules indicating which controllers can execute tasks in parallel and which should be strictly sequential.

The results of the data acquisition are stored into several different file formats. ASCII data files contain all the relevant information organized in a text format following specific rules to be compatible with the existing ILL's data treatment programs. NeXus [3] files contain binary and compressed data. Nexus is a common data format share by neutron, x-ray and muon facilities. It is based on the Hierarchical Data Format (HDF) developed by U.S laboratories and extremely powerful for large data sets. LIST-MODE files are pure binary containing, beside a specific ILL header, the sequence of all detected events and the associate time-stamp to provide a maximum of flexibility to replay the experiment off-line.

NOMAD records all important instrument history information in daily log files. The ASCII version is directly accessible by the user via his favourite text editor. The server includes in this file some additional debugging information that can be helpful for the developers in case of unexpected behaviour. NOMAD is also saving a XML version of the log file. This can be accessed by the NOMAD client application and provides additional features to the user like filtering on specific actions and calendar navigation.

All instrument parameters are permanently monitored and stored into daily Survey XML files. Those data are displayed by the NOMAD client application in a dedicated plot window and the user is able to dynamically select and visualise a specific set of system's variable.

The server automatically notifies to the administrator or to the users all errors occurred during the execution.

Client

The Nomad Client application written in Java [4] SWT is designed to offer a practical Graphical User Interface (GUI) for the user to control the instrument. A dedicated effort has been made to provide an easy-to-use graphical programming of complex sequences of instructions. The NOMAD client application consists of three different graphical views specifically designed to satisfy different needs. The *Hardware* view is intended for the technical support and gives access to all hardware specific parameters like displacement limits, positioning speed, etc… The *Setting* view is mainly devoted to instrument responsible and allows configuring higher-level controllers. Finally the *Launch-Pad* view permits the creation and the execution of a specific workflow.

The *NomadCommandSystem* library is a fundamental component of the client architecture. It is designed as a bridge between the client and the server. It abstracts the way to request the server and it mainly uses Corba proxy objects to communicate with server. It implements some Corba IDL interfaces since it also acts as a server from the Corba point of view.

To minimise the coding work we have designed an XML file format to describe controllers and drivers' GUI without writing any Java SWT code. However, some specific GUI behaviours that cannot be obtained by XML description are directly coded in Java SWT. Some additional configurations files are required by the client application to start. They concern for example label text properties.

Nomad Monitor

We have designed a NOMAD Monitor application written in Java RCP to visualise the actual status of the NOMAD Server system. The application offers the list of the controllers and drivers showing all their property and

the associated values. Controller and driver objects are loaded by requesting the Nomad Server. Nomad Monitor enables debugging controller and driver execution and acts also a configuration manager giving the possibility to dynamically add a new controller or a new driver while the server is running.

Web Spy Server

To provide the user with a maximum of information to promptly react in case of a problem or simply to remotely verify the status of the measurement we have developed a web server that retrieves periodically from NOMAD Server all the actual values of properties. An instrument specific web page is then updated to provide all important information on the instrument (see Fig. 2).

Figure 2: Layout of the web spy for the instrument IN4.

INTER-APPLICATION COMMUNICATION

The CORBA [5] architecture has been chosen to make the connection between the C++ server and the Java client. The CORBA standard offers a flexible interoperability service through the definition of IDL interfaces and its implementations provide good performances in regard of other inter-application communication like, for instance, Web services. However we made the choice to minimize the dependency to CORBA by avoiding the number of CORBA objects for which life cycle is not so easy to manage in C++ and leave them only to the communication layer. For this reason, all our CORBA objects are wrapped into Adapter objects (client and server) that we call *Accessors*. Server and clients use a two-way communication for:

- access/execute: the clients request the server to get or set a value, to execute tasks, etc…
- provider: to provide the actual values of the variables of the system (e.g. the properties of the instrument)
- notify: the server notifies the client when a server variable has changed.

For the client to server requests (on a user request), the client contacts directly the server through the CORBA *Accessor* objects. The server to client request uses the *Observer* pattern to implement an event-handling system.

NOMAD server registers CORBA subscriber objects to dispatch events to them. The server and the clients both use a Producer/Consumer pattern for respectively send and process the event, resulting in an asynchronous event notification. Figure 3 shows a simplified example of server to client forward and backward communication. The Access Data model illustrate the different patterns, synchronous and asynchronous messages from the notification of the server of a value changed to the call of the client for getting the value.

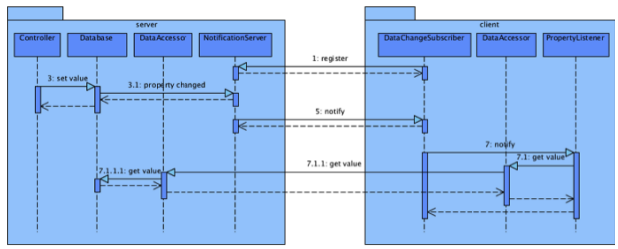


Figure 3: Data Access sequence.

The distributed Publisher/Subscriber objects when the client and the server communicate over a network offers great advantages to refresh the client only when needed and prevent from using client polling, but one drawback is that we can face client firewall port denials.

THE NOMAD SERVER LAYERS

The NOMAD server application is a multi-threaded application designed in three distinct layers.

- Data Provider is the part of the server that stores the data model, e.g. the hierarchy of controllers and drivers with their associated properties. We also store the list of commands accessible from each controller and driver and their current state. This is the data representation of the controllers and drivers running in NOMAD server for a specific instrument (loaded from the configuration files).
- Command can be considered as the business logic layer. It is responsible to execute the requested commands to the controllers and drivers. Each controller or driver accesses its property data from the Data Provider layer. We have designed a framework of classes for controller and drivers to facilitate the business code integration. We have chosen to implement an event-driven programming [6] model to react easily to the real instrument dynamic behaviour (listening to sensors). *AbstractController* is the abstract class to specialize. It contains access methods to associated properties that are bridges to the Data Provider data model. It implements the methods *execute*, *refreshSetValue*, *updateProperty*. The method *execute* contains the execution code of the different registered commands. The most important command is *start*. The *refreshSetValue* is the function implementing how the controller reacts when one of its properties change. This is the way to spread information from the high-level layer (controller) to the low-level layer (driver). The *updateProperty* is the function

describing how the controller reacts to the change of a property of one of its linked controllers or drivers. This is the way to spread information from the low-level layer to the high-level layer. NOMAD has at present about 250 controllers (from the most general to the most instrument specific) and 150 drivers available.

Error management controls the way NOMAD server reacts to the real behaviour of the instrument that can be seen as a complex system depending on many uncontrolled parameters. During the development phase we have rarely access to the real instrument therefore, all the tests performed in the lab cannot be exhaustive. To limit error's occurrence we run logical unitary and functional tests with a basic simulation of the system. A second series of tests aim the verification of the consistency of all instrument configurations. Unexpected behaviours can lead to exceptions, reported in the log files, or to crashes. The last are notified by mail to the developer's team together with the maximum of available information to allow further analysis of the problem. Errors connected to defective behaviour of the instrument generate a series of alarms or warnings that are notify by mail to the instrument responsible.

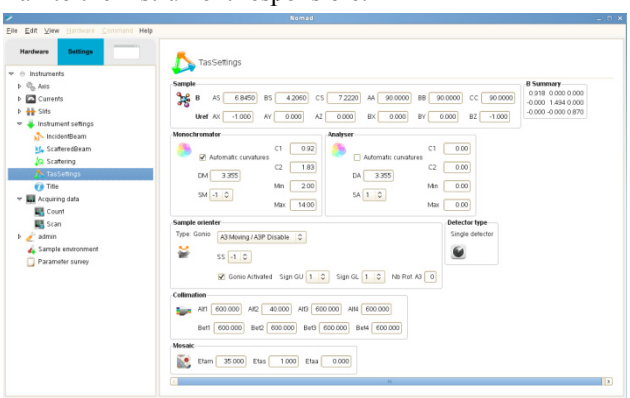


Figure 4: An example of abstract controller.

INSTRUMENT ABSTRACTION

Abstract controllers are used to hide from users the specific underlying hardware or control sequences. It provides an interface relevant to a specific function and then passes user requests to the more specific instances.

The main purpose of the instrument control software renewal is to free human resources and re-uses them to integrate prototyped versions of scientific methods into NOMAD. The main objective is to perform faster and more accurate measurement through intelligent scientific methods and thus increase the efficiency of instruments.

Handling instrument methods complexity, instruments control methods appears at first glance as complex and of large variety. The scientific analysis shows that instrument and scientific methods can be well organized into common elements with specific parameterization.

They are organized in term of primary and secondary spectrometer, sample orienteer, basic scattering and orientation strategies. Presently, the design of the scientific methods shows that only a few set of objects are actually needed. The present complexity and variety comes from the fact that mathematical models are merged into single routines, rigidifying the code. As well, science groups have different notations referring to the same scientific concepts. These will be handled by the GUI, letting the physics concepts behind untouched. Fig. 4 depicts and example of abstract controller that regroup in one single interface all the quantities needed to properly initialise a three-axis spectrometer and prepare it for the experiment. Complex instrument's setup but also alignment and acquisition sequences have been optimized and specifically coded in NOMAD to get the best out of the available beam-time and to facilitate users' tasks.

CONCLUSION

After an initial phase of heavy debugging, NOMAD is meanwhile installed on all ILL's instruments for motors and general hardware setup. So far, more than 20 instruments are fully controlled with it and more are added on a regular base. In addition, 10 installations have been performed to drive technical equipments within the Technical and Project Department. NOMAD includes meanwhile all the hardware devices present at the ILL and all the generic sample environment modules as well as all the instruments' specific ones. It provides the users the freedom to configure dynamically the equipment needed for a specific experiment. The full integration between sequencer, hardware devices and sample environment allows a much better event handling and full synchronization. As well, physical quantities, like energies or wavelength, are compounds of several hardware components. The control of the instrument directly through physical values, those who have a meaning for the user, implied a new way of managing interconnected hardware pieces. Giving control at a more physical than hardware level is in the spirit of opening our instruments suite to a broader audience.

REFERENCES

[1] G. Reese, "Database Programming with JDBC and Java" (2009); htth:// java.sun.com/developers/Books/jdbc.
[2] B. Stroustrup, "The C++ Programming Language", Addison-Wesley Professional (2000).
[3] http://www.nexusformat.org.
[4] K. Arnold, J. Gosling and D. Holmes, "The Java Programming Language", 3rd Edition Addison-Wesley Professional (2000).
[5] S. Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments", IEEE Communications Magazine, Vol. 14, No. 2 (1997).
[6] S. Ferg, "Event-Driven Programming: Introduction, Tutorial, History" (2006); http://Tutorial _EventDrivenProgramming.sourceforge.net

AUTOMATIC CREATION OF LabVIEW NETWORK SHARED VARIABLES

Thomas Kluge, Siemens AG, Erlangen, Germany
Harm Schroeder, ASTRUM IT GmbH, Erlangen, Germany

Abstract

We are in the process of preparing the LabVIEW controlled system components of our Solid State Direct Drive® experiments [1, 2, 3, 4] for the integration into a Supervisory Control And Data Acquisition (SCADA) or distributed control system. The predetermined route to this is the generation of LabVIEW network shared variables that can easily be exported by LabVIEW to the SCADA system using OLE for Process Control (OPC) or other means. Many repetitive tasks are associated with the creation of the shared variables and the required code. We are introducing an efficient and inexpensive procedure that automatically creates shared variable libraries and sets default values for the shared variables. Furthermore, LabVIEW controls are created that are used for managing the connection to the shared variable inside the LabVIEW code operating on the shared variables. The procedure takes as input an XML spreadsheet defining the required input. The procedure utilizes XSLT and LabVIEW scripting. In a later state of the project the code generation can be expanded to also create code and configuration files that will become necessary in order to access the shared variables from the SCADA system of choice.

PROBLEM INTRODUCTION

Using LabVIEW [5] network shared variables [6] makes it easy to connect LabVIEW controlled hardware components to a SCADA or distributed control systems via OPC [8] (see Fig. fig:ExVarDef). You have to provide a set of network shared variables and LabVIEW code that reacts on changes to these variables or updates them. LabVIEW's shared variable engine provides the OPC server.

However, when creating and using shared variable libraries in LabVIEW we've been facing the following tasks that require similar or even identical actions or coding for each variable.

- creating a shared variable library,
- creating a shared variable,
- initializing the value of a new shared variable,
- opening and closing the connection to a shared variable,
- reading and writing shared variable values.

Later, when the connection to a SCADA or distributed control system has to be done, for each library configuration files will have to be written that describe the shared variable libraries for the integration into the system. With multiple instances of the same hardware component the variable creation tasks are multiplied with the number of instances.

The effort for this tasks is significant as it is comparable to the implementation of the real functionality of the code that connects the shared variables with the hardware driver calls.

MATERIALS AND METHODS

Shared Variable Library Description

A shared variable library is identified by a location URL and a name. The library can contain variables of simple types (integer, floating point, boolean, string) and one-dimensional arrays of simple types. Each variable has a unique name inside the library. For each variable we want to define a default value that can be used for initialization of newly created variables.

Required LabVIEW Clusters

For the generic LabVIEW functionality described later we need a set of LabVIEW clusters for each shared variable library. The clusters for each shared variable in the library contain one element named identically to the shared variable in order to store specific information.

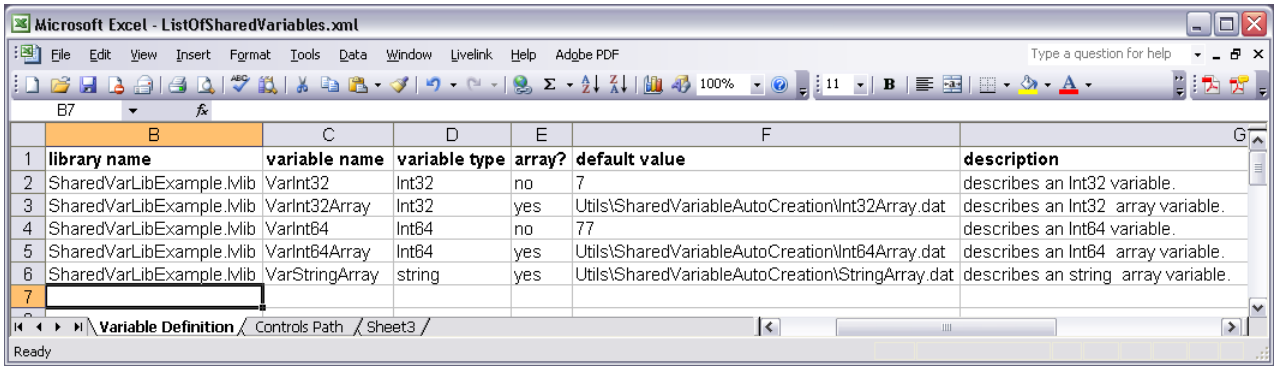
- The elements of the "type cluster" have the same type as the shared variable. The "type cluster" can be used for getting information about the data types of the shared variables and for storing values of the shared variables.
- The elements of the "refnum cluster" are LabVIEW's shared variable refnums. They are used for maintaining information about open connections to shared variables.
- The elements of the "access cluster" are booleans. They are used for controlling which shared variables are accessed by generic LabVIEW functionality.

Generic LabVIEW Functionality

Using references to the LabVIEW controls just introduced the following functionality can be provided by generic LabVIEW Virtual Instruments (VI) for all or only a subset of the shared variables inside a library

- creating a variable,
- opening the connection to a variable,
- closing the connection to a variable,
- reading values,
- writing values,
- pre-setting access control cluster.

Shared variable library specific VIs providing this functionality for the clients of the shared variale library just use references to the specific clusters and call these generic VIs.



	B	C	D	E	F	G
1	**library name**	**variable name**	**variable type**	**array?**	**default value**	**description**
2	SharedVarLibExample.lvlib	VarInt32	Int32	no	7	describes an Int32 variable.
3	SharedVarLibExample.lvlib	VarInt32Array	Int32	yes	Utils\SharedVariableAutoCreation\Int32Array.dat	describes an Int32 array variable.
4	SharedVarLibExample.lvlib	VarInt64	Int64	no	77	describes an Int64 variable.
5	SharedVarLibExample.lvlib	VarInt64Array	Int64	yes	Utils\SharedVariableAutoCreation\Int64Array.dat	describes an Int64 array variable.
6	SharedVarLibExample.lvlib	VarStringArray	string	yes	Utils\SharedVariableAutoCreation\StringArray.dat	describes an string array variable.
7						

Figure 1: Example of a shared variable library definition. The XML spreadsheet can be edited with Microsoft's Excel.

Automatic Code Generation

The information about the shared variables and the names of the generated artifacts are stored in an XML file [10] that conforms to the Microsoft XML Spreadsheet [13] schema [12]. This format makes it easy to edit the information and allows us to process it with XSLT stylesheets [11]. The XML file is XSLT-transformed into an XML string that conforms to the LVData XML schema [7] and is then converted into a LabVIEW cluster. This cluster is used by LabVIEW scripting [9] for the generation of all shared variable library specific clusters and VIs.

EXAMPLE

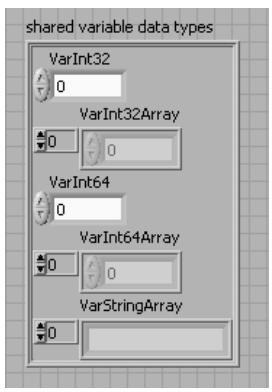


Figure 2: Example of an auto-generated "type cluster".

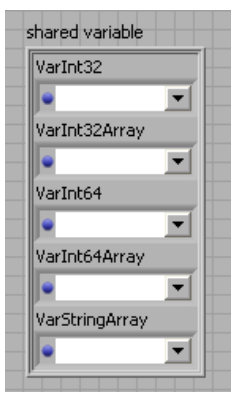


Figure 3: Example of an auto-generated "refnum cluster".

Figure 4: Example of an auto-generated "access cluster".

NEXT STEPS

The tool-chain we have introduced will be used in a next step for decoupling the LabVIEW graphical user interfaces from the underlying hardware control functionality. After this has been done the hardware functionality will be connected to a SCADA system. The tool-chain will be expanded in order to generate the SCADA system specific configuration files for the shared variable libraries.

SUMMARY

LabVIEW scripting has successfully been utilized for LabVIEW code generation allowing significant reduction of recurring work for generating and using LabVIEW shared variable libraries. The LabVIEW-independent XML input information can easily be utilized for further text-base code generation within the build process.

Figure 1 shows a screen-shot of the spreadsheet defining a shared variable library. The auto-generated LabVIEW clusters are shown in Figures 2, 3 and 4. Figure 5 gives you an impression of a generic VI that operates on the auto-generated clusters. The VI shown in Figure 6 is auto-generated and calls the generic VI with references to specific cluster instances. Finally, Figures 7 and 8 illustrate the use of LabVIEW scripting for creation of the cluster shown in Figure 2.

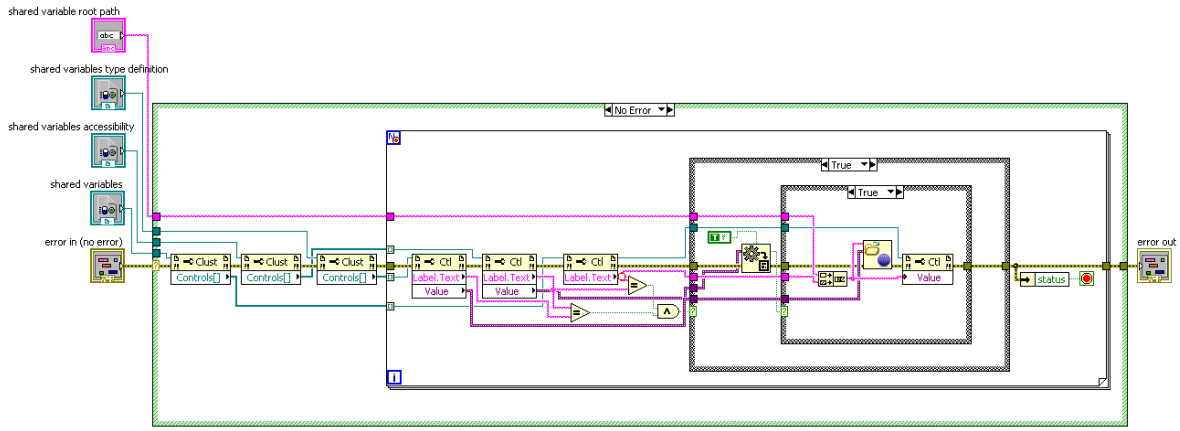


Figure 5: Example of a generic VI. The block diagram shows how shared variables are opened using the auto-generated clusters.

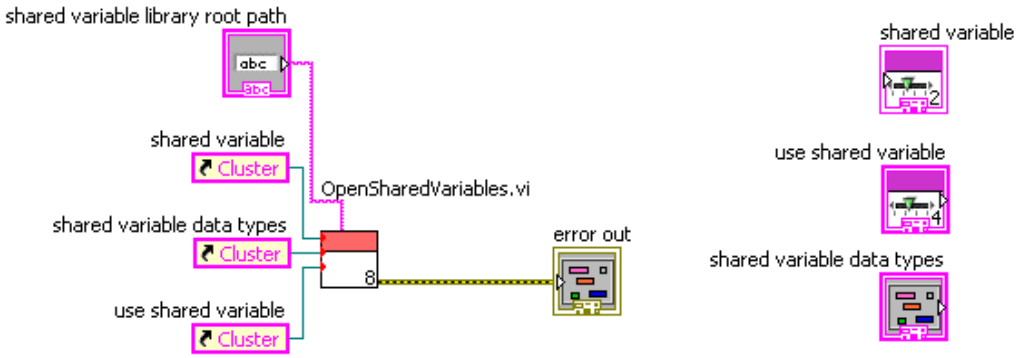


Figure 6: Example of an auto-generated VI that uses the generic VI from Figure 5 for opening shared variables from a specific library.

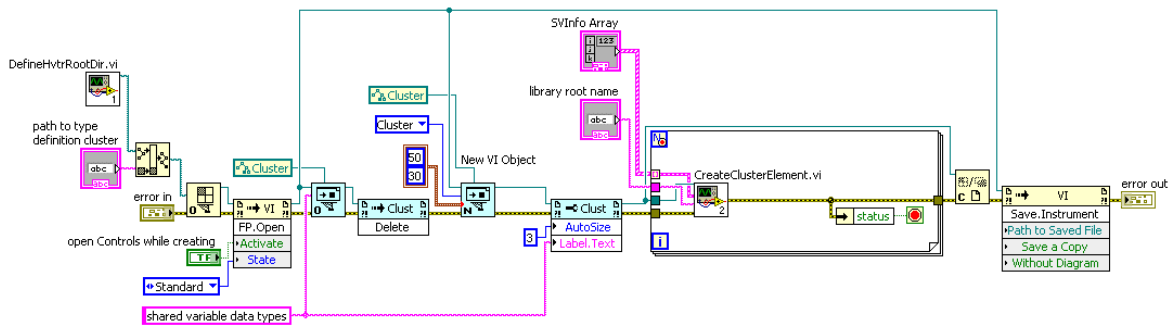


Figure 7: Example of a LabVIEW scripting VI. This VI creates the cluster shown in Figure 2. The sub-VI CreateClusterElement.vi is shown in Figure 8.

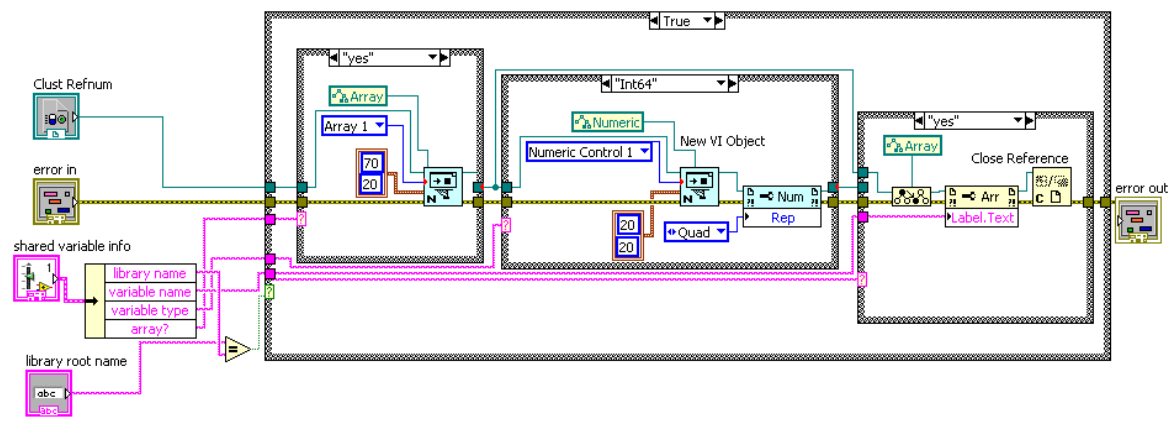


Figure 8: Example of a LabVIEW scripting VI. This VI creates the elements of the cluster shown in Figure 2.

REFERENCES

[1] O. Heid, T. Hughes, THPD002, IPAC10, Kyoto, Japan

[2] R. Irsigler, et al, 3B-9, PPC11, Chicago IL, USA

[3] O. Heid, T. Hughes, THP068, LINAC10, Tsukuba, Japan

[4] O. Heid, T. Hughes, MOPD42, HB2010, Morschach, Switzerland

[5] National Instruments LabVIEW, http://www.ni.com/labview

[6] Using the LabVIEW Shared Variable, http://zone.ni.com/devzone/cda/tut/p/id/4679

[7] LVData XML Schema, LVXMLSchema.xsd comes with LabVIEW installation in vi.lib/Utility sub-folder

[8] The OPC Foundation, http://www.opcfoundation.org/

[9] LabVIEW Scripting, https://decibel.ni.com/content/docs/DOC-4973

[10] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml

[11] XSL Transformations (XSLT), http://www.w3.org/TR/xslt

[12] XML Schema, http://www.w3.org/TR/xmlschema-1 and http://www.w3.org/TR/xmlschema-2

[13] Microsoft Office 2003 SpreadsheetML, http://msdn.microsoft.com/de-de/library/aa140066.aspx and http://www.microsoft.com/download/en/details.aspx?displaylang=en\&id=101

SOFTWARE FOR VIRTUAL ACCELERATOR DESIGNING

N. Kulabukhova, A. Ivanov, V. Korkhov, A. Lazarev, SPbSU, Saint-Petersburg, Russia

Abstract

The article discusses appropriate technologies for software implementation of the Virtual Accelerator. The Virtual Accelerator is considered as a set of services and tools enabling transparent execution of computational software for modeling beam dynamics in accelerators on distributed computing resources. Distributed storage and information processing facilities utilized by the Virtual Accelerator make use of the Service-Oriented Architecture (SOA) according to a cloud computing paradigm. Control system toolkits (such as EPICS, TANGO), computing modules (including high-performance computing), realization of the GUI with existing frameworks and visualization of the data are discussed in the paper. The presented research consists of software analysis for realization of interaction between all levels of the Virtual Accelerator and some samples of middleware implementation. A set of the servers and clusters at St.-Petersburg State University form the infrastructure of the computing environment for Virtual Accelerator design. Usage of component-oriented technology for realization of Virtual Accelerator levels interaction is proposed. The article concludes with an overview and substantiation of a choice of technologies that will be used for design and implementation of the Virtual Accelerator.

INTRODUCTION

The key idea of Virtual Accelerator (VA) concept is beam dynamic modeling by the set of several packages, such as COSY Infinity, MAD, etc., based on distributed computational resources, organized on Grid-technology.

Simulation beam dynamics by different packages with the opportunity to match the results (in case of using different resources for the same task) and the possibility to create the set of tasks when the results of using one package can be sent to the input of another is the main use of VA.

Users will get the access to VA resources by unified interface including GUI on different platforms:

- Web-based application: web-services, web browsers' application (Java Webstar technology, Silverlight, etc.).
- Desktop applications: Windows, Linux.
- Mobile platforms: Android, Windows Phone 7.

Note that the VA is considered as information and calculation environment and does not refer to the real-time control systems. On the other hand this control organization must be provided by connection to the specialized software (e.g. EPICS). Such kind of VA is examined in [1] and [2] where authors emphasize on accelerator control development.

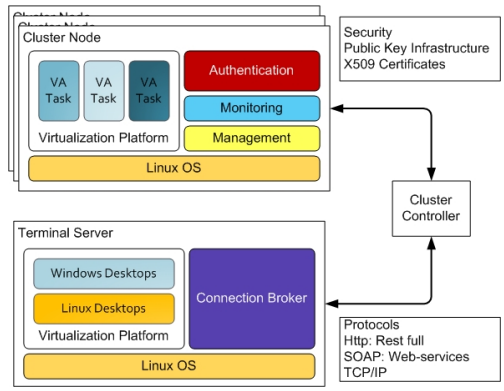


Figure 1: Virtual Accelerator.

In Fig. 1 VA scheme is shown. Cluster controller is a middleware framework developed to interact with calculation programs. This framework provides access by widely used standard protocols such as HTTP, SOAP, TCP/IP.

The same approach to develop a virtual laboratory is discussed in papers [3] and [4] for nuclear physics applications. In article [5] a heterogeneous computing environment and development of the distributed computing systems development are examined.

VIRTUAL ACCELERATOR SCHEME

The usage of VA as a computational resource can be presented as following:

1. Access to the VA resources by the authorization and authentication of users.
2. Specify initial data and settings for calculation:
   - on generalized language for system description (e.g. XML-based),
   - on native package language (COSY Infinity, MAD, OptiM).
3. Choice of the packages and simulation algorithms. Note that user can define the structure on one language (for example, on COSY Infinity notation) and select another program for calculation (e.g. OptiM). In this case conservation of system description will be made by the VA infrastructure invisible for user. And giving the instructions to start the simulation.
4. Monitoring of the task states, data visualization and representation. Access to a logging system with error tracking.

5. Access to the storage of user data that can be represented as simple remote file system with users directories and files. WebDAV protocol can be used for such system.

Security: Access to VA

The VA is available for users as Cloud-service (based on Grid-technology) with standart security features. The authentication techniques are:

- Username and Password
- Public Key Infrastructure and X509 Certificates

Transport level security is provided via SSL encryption. For the prototyping and testing we use OpenSSL program to generate own certificates.

Middleware Implementation: Server Side

Nowadays there exist number of programming languages and technologies. For VA implementation we use Java VM and .NET Framework[1]. Each of these platforms has great built-in capabilities for developing server side software. For example, WCF (Windows Communication Foundations) as a part of .NET Framework provide mapping of business logic in high-level programming languages into web-protocols without any additional effort other than adding declarative information to existing methods.

Client Application

As we mentioned above VA can provide unified access to it services both for web and desktop application. Not

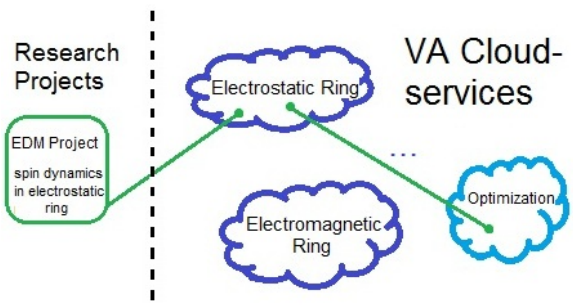


Figure 2: Cloud-services for EDM-project.

all services are required in each case. For example, for EDM-search project[2] it is necessary to include only electrostatic elements, optimization modules and related components (see Fig. 2). Other modules are enabled but available for the research purposes.

An example of the program for EDM project is presented on Fig. 3. It involves electrostatic LEGO objects (elements

[1]Microsoft .NET Framework under Academic Alliance License and Mono as OpenSource platform.

[2]St. Petersburg State University collaborates with Institute for Nuclear Research of Forschungszentrum Jülich, Germany.

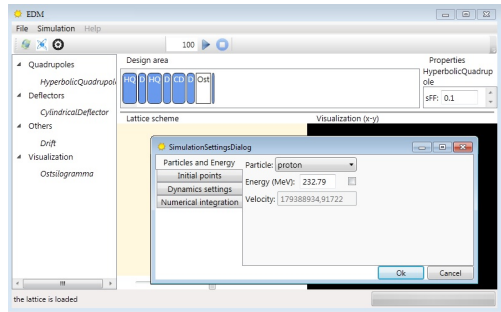


Figure 3: Windows desktop application.

with its parameters, fringe field components), optimization module (e.g. optimization module based on genetic algorithm approach) and visualization tools.

Task Monitoring

For the calculation complex problems (such as long-term beam evaluation) it is important to provide monitoring tools. Researchers at any time can have an opportunity to see progress of the task, restart or cancel it, save obtaining result and etc. VA services must provide real-time interaction with users requests. This requirements provide a complete integrated computing environment for composing,, running, controlling and visualizing applications.

Connection to Physical Equipment

In VA concept it is common interface for simulation model and there should not be differences between access. Both simulation model and equipment adapter (e.g. under the EPICS) must implement the same interfaces. EPICS can be used as a part of the simulation process.

The common user interface of VA allows us to get solutions both in simulation models and in a real machines. This approach provides researcher some mechanism of system identification, parameters optimization and result verification. Such opportunity is impossible without computational models and is a goal aim of the current research.

LEGO PARADIGM IMPLEMENTATION

A LEGO paradigm for VA design is described in [6]. In terms of information technology it corresponds to object oriented design and component programming. Each object represents as independent component with own parameters and behavior. Building the proper class hierarchy allows developer to scale and modify the objects using the already existing solutions. C# and .NET are used for prototyping at the moment. This technology has several advantages compared to Java VM (pointers and memory allocation in the stack, declarative description of the desired functionality, etc.). But in general, these platforms are very similar and the choice depends on the preferences of the development team at all. Note that for implementation of numerical al-

Table 1: Summary of Used Languages

Language	Performance	Web Interfaces	Flexibility
C++	++	--	--
C#	+-	++	++
Java	-+	++	+-

gorithms it will be use C++ as the most powerful language for computing tasks.

A LEGO object is represented as a component object. In object oriented design it means a class with predefined interfaces. Objects has an inheritance hierarchy in that LEGO objects can defined.

LEGO means an ability to choose and change necessary elements without reconstruction of whole structure. For example, in simulation we have a set of integration methods. Of course, each of them mast be used in specific cases that depends on physic problem, mathematical requirements and etc. But for LEGO paradigm they are all one object that knows how to obtain solution and to present it in some form.

SCIENTIFIC VISUALIZATION

Scientific visualization plays an important role in result processing. In the images and animations researcher can diagnose some features associated with the intuitive understanding of the physics of the process rather than they can obtain the same result during statistical data processing.

Visual representation of data help to verify obtaining result and to quickly detect errors that invisible for calculation algorithms but critical to plausibility checking.

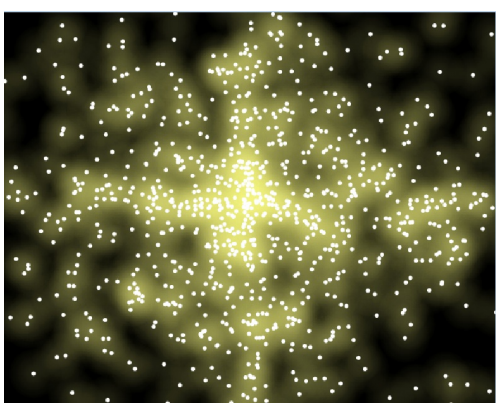


Figure 4: Coordinatewise slice of beam.

In Fig. 4 a coordinatewise ($x - y$ space) slice of beam is shown. In this figure an initial particle distribution is presented. On the picture we can see some clusters of points that may be difficult to find by data mining methods.

CONCLUSION

In the article a scheme of VA organization is described. Data access protocols that will be used are mentioned. Some modules such as global optimization tools, simulation and numerical algorithm are completely developed, other are in a progress. Future plans for this project include extend of the VA tools, implementation of Cloud-service architecture and support developed systems via knowledge base growing, development of methods and approaches in beam dynamics and simulation and validation proposed approach on real machines.

Some approaches that was described above were tested in the distributed computational environment at the faculty of Applied Mathematics and Control Processes on the department of Computer Modelling and Multiprocessor Systems[3].

REFERENCES

[1] P.C. Chiu, C.H.Kuo, Jenny Chen, Y.S. Cheng, C.Y.Wu, Y.K.Chen, K.T. Hsu, "Virtual Accelerator Development For The TPS", IPAC'10, Kyoto, Japan 2010, WEPEB019, p. 2728, http://www.JACoW.org.

[2] C. Gulliford, I. Bazarov, J. Dobbins, R. Talman, N. Malitsky, "The NTMAT EPICS_DDS Virtual Accelerator For The Cornel ERL Injector", IPAC'10, Kyoto, Japan 2010, WEPEB022, p. 2734, http://www.JACoW.org.

[3] V. Korkhov, D. Vasyunin, A. Belloum, S. Andrianov, A. Bogdanov, "Virtual Laboratory and Scientific Workflow Management on the Grid for Nuclear Physics Applications", Distributed Computing and Grid-Technologies in Science and Education: Proc. of the 4th Intern, Dubna, Russia 2010, p. 153.

[4] A. Belloum, D. Groep, et al., "VLAM-G: a grid-based virtual laboratory", Future Generation Computer Systems, V. 19, 2003, P.209–217. Distributed Computing and Grid-Technologies in Science and Education: Proc. of the 4th Intern, Dubna, Russia 2010, p. 153.

[5] A. Bogdanov, A. Lazarev, Myo Tun Tun, La Min Htut, "Development of the Distributed Computing Systems and Running Applications in the Heterogeneous Computing Environment", Distributed Computing and Grid-Technologies in Science and Education: Proc. of the 4th Intern, Dubna, Russia 2010, p. 69.

[6] S. Andrianov, A. Ivanov, E. Podzyvalov, "A LEGO Paradigm For Virtual Accelerator Cocept", in this Proc.

[3] http://www.apmath.spbu.ru/en/staff/andrianov/

MSTAPP, A RICH CLIENT CONTROL APPLICATIONS FRAMEWORK AT DESY

Kirsten Hinsch, Winfried Schütte, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

The control systems for PETRA 3 [1] and its pre accelerators extensively use rich clients for the control room and the servers. Most of them are written with the help of a rich client Java framework: MstApp. They total to 106 different console and 158 individual server applications. MstApp takes care of many common control system application aspects beyond communication. MstApp provides a common look and feel: core menu items, a colour scheme for standard states of hardware components and predefined standardized screen sizes/locations. It interfaces our console application manager (CAM) [2] and displays on demand our communication link diagnostics tools [3], [4]. MstApp supplies an accelerator context for each application; it handles printing, logging, resizing and unexpected application crashes. Due to our standardized deploy process [5] MstApp applications know their individual developers and can even send them – on button press of the users - emails. Further a concept of different operation modes is implemented: view only, operating and expert use [6]. Administration of the corresponding rights is done via web access of a database server. Initialization files on a web server are instantiated as JAVA objects with the help of the Java SE XMLDecoder [7]. Data tables are read with the same mechanism. New MstApp applications can easily be created with in house wizards like the NewProjectWizard [5] or the DeviceServerWizard [8]. MstApp improves the operator experience, application developer productivity and delivered software quality.

INTRODUCTION

As part of the transition of PETRA 2 - an injector for HERA - to PETRA 3 - a dedicated synchrotron machine - an entire new control system was implemented [1], [9]. It is based on Java as a platform independent language, TINE [10] as a communication protocol and MstApp as an application framework.

The control system tasks are implemented with networked rich client JAVA SE 6 applications. We distinguish between console and server applications. And for some complex cases we also have middle layer applications.

Many aspects of these applications are generic. A framework called MstApp was designed to cover the generic aspects not related to the communication protocol. The application developers could then concentrate on the application specific aspects.

LOOK AND FEEL

A common look and feel of all accelerator control client applications eases the task of the operating crew. This is both true on the small level of standard colours for different component states, and on the large level like the layout of the entire application. Printing, protocols, help etc. are always found at the same place.

Standard Colours

There are always a lot of discussions about the best colours for different component states. And once there is an agreement how to find the documentation. The framework MstApp just integrates our agreement on colours as mnemonically named subclasses of the standard Java "Color" class. As an example the colour for error states is defined as a red background with a white foreground (a black foreground would be too difficult to read on a red background).

Standard Sizes and Positions

Applications are not alone on the display but have to live together with many other ones. Well defined application sizes help to use the screen better. MstApp gives predefined sizes and positions. Both can also be saved on a local basis to the hard disc or managed by our console application manager (CAM) [2]. This feature is useful for the control room by having standard work environments. It is also useful for server PCs with many servers. A missing small server display is spotted at a glance.

Standard Menu

Some standard functionality is already provided in the menus. The predefined layout is the following:

- File, containing show logging, printing and exiting
- Machine for the applicable accelerator (optional).
- Options, containing an extensive dialog
- Help, containing an about box, and paths to the most relevant pages of the accelerator control room WIKI

The application developer can add new menus, or in existing menus new menu items. The options dialog is also extensible.

Resizing

A high fraction of our application developers like to design with absolute layout. No layout manager is used. Since the size of our applications is related to the screen size, this works well as long as the size of our display monitors does not change. In reaction to the unavoidable change of display size, a so called ApplicationResizeManager was developed.

An activated ApplicationResizeManager resizes all components of the application top-down. The following properties are resized: position, width, height, font, tables and icons. AWT graphics has to be scaled manually in the paint method. Resizing the font the ApplicationResizeMa-

nager considers both, width and height of the component to calculate the new font size.

Auxiliary Information

A standard MstApp application contains at the bottom a status bar with information about the PC name, date, time, operations and server mode. This is especially helpful for graphical application copies to the logbook [11].

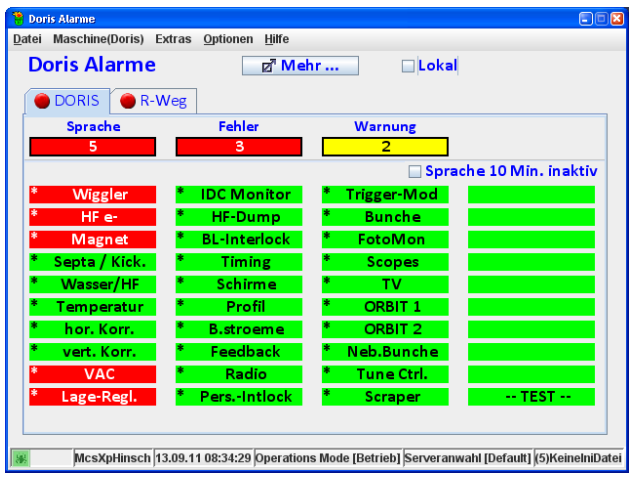


Figure 1: Typical MstApp based client application. The accelerator Doris is chosen. An application developer specific menu “Extras” is added. The status bar at the bottom shows the computer name, date and operations mode. At its left one finds the communication link diagnostic tool Spider.

COMMON TASKS

Operation Modes

Console programs for accelerator control have different users.

Our most important users are the operators in the control room. The operator needs good reading and writing access to the servers running the accelerator hardware.

Then there are users outside the control room interested in current accelerator parameters. They should not be allowed to change the accelerator hardware parameters. Their program use should be restricted to only viewing them.

Finally there are the hardware developer themselves, who need to access their specific devices in an expert way reading and writing to the corresponding servers far in excess of the operators.

Each of these views operating, viewing and expert mode is natively implemented in the framework. Maximum rights are enforced on a basis of user name, pc name and application name via a database table.

We also differentiate between console application accesses to the real hardware server and a simulation server.

These security measures are complementary to the ones inherent to the communication protocol. See for example [12].

Accelerator Context

There are applications so specific that they run only for a single accelerator. Others like the display of the dc current monitor are the same for all circular accelerators. We supply a context for all those accelerators and a standard switch mechanism between them.

Printing to Logbook and Developer E-Mails

The MstApp framework provides an identical print dialog frame for each application. This frame enables the users to print a screenshot to a normal printer as well as to create full logbook entries [11]. The user has a mask for his name, a category and some additional text. A screenshot of the application is added automatically. The user can print this entry to the accelerators logbook (default) and/or sent it as an E-Mail to the developer. To declare his choice he uses two checkboxes.

Help

Help is a very difficult topic. There has to be help at all, it has to be up to date and correct, and the user has to be able to find it. Some users might even want to improve on it.

Due to our common deploy process [5] a minimal up to date help within the application is possible. In this about box style help the user finds the application developers name, telephone number and email address. He finds the version, creation date and location of the application and all its referenced libraries.

If more help seems to be useful, the developer can register a page of our control room wiki. This will be offered by the framework to the user. Still someone has to create and maintain this page.

Console Application Manager (CAM)

For routine accelerator operation it is helpful to have a defined set of applications with defined places for the consoles. Our console application manager [2] does this for us. All MstApp applications are native CAM clients. No developer has to invest here any work.

Communication Link Diagnostics Tools

Network communications is never absolutely reliable. It is therefore mandatory to have some handy analysis tools for an application. The Spider [4] shows all direct TINE [10] links and Tarantula [3] shows the direct links, the links of the linked servers and so on up to a specified depth.

Unhandled Crashes

Obviously applications should never end abnormally (crash). In practice they do crash on some rare occasions. In this case it is important to end the application gracefully and to help the developer to improve on the code.

Our framework tries to ease on this. A Java shutdown hook catches those uncatched errors and tries to display a crash screen. The crash screen allows for printing a screen shot, the about box information and a comment to the accelerator control logbook. Also an email can be sent to the known application developer simple by the press of a button.

Applications do not only die by crashing. Sometimes they just freeze. This is especially bad for server programs. Here a watchdog [2] might stop and restart the server application. A screenshot is put into the TINE protocol server. All those screenshots can be viewed from any framework application.

Initialisation Files

The framework supplies a platform independent way of accessing files from a webserver. Initialisation files, some data tables and the data for allowed operation modes are transported in an xml encoded fashion that can be directly converted to a tree of Java objects with the help of the Java SE XMLDecoder [7].

Technically this works very well though the actual use is a little cumbersome.

Protocols/Logging

Writing protocols for accelerator applications has different objectives than the standard Java logging tools. Therefore a custom logging mechanism for application protocols was introduced. The last 500 (can be configured) entries are kept in main memory and the last seven days are kept on local disc for display directly from the application. Logging from the framework and the application and caught logging from the Java logging mechanism is tagged with a category. Logging of Java exceptions is also supported. By supplying a tag the logging message will be stored in the central TINE logger server [13].

STANDARD SERVER APPLICATIONS

The server framework provides a special JFrame for server applications. It ensures an identical look and feel. So the developers can concentrate on the work without graphical aspects. The server frame inherits from an MstApp client frame and provides all aspects the client has.

A server application has three resize modes. Normally it is small, just a square with the name, the icon and a state like idle or reading data. The small size allows many applications neatly aligned in rows.

Pressing the maximize button the application resizes to medium size. Here additional output shows the start time, the last commands and the last events. The entries to two lists for the commands and the events are done with the MstApp logging commands by using special category tags.

In expert mode the application resizes to the large size. It shows a pane, to which the developer is free to add test buttons and output features, which are interesting for the particular server. Other parts of the frame should not be changed.

The non-graphical work of the server application is coded in classes created by the DeviceServerWizard [8]. The developer adds device specific code in his own custom classes.

Figure 2: A server in small size. It only shows the name, the menu and an icon. The status bar is hidden.

INTEGRATION WITH OTHER FRAMEWORKS AND WIZARDS

Our framework interacts by design well with our other tools: The NewProjectWizard [5], our standard deployment [5] and the DeviceServerWizard [8]. It works with the communication protocols like TINE [10] and DOOCS [14]. Even the combination with jddd [15] is easily possible.

TECHNICAL NOTES

How is it realized in Java? The developer has to supply a special Java class that inherits from a special JFrame: MstFrameMain or in case of a server MstServerFrameMain. MstFrameMain supplies context information and the basic look and feel. Components from the developers are added only to the central user area. The framework creates the MstFrameMain and knows it via a "String" starting parameter. Also parameters for the application name, the accelerator name und application specific parameters can be supplied.

The logging is supplied with a static mechanism.

Standard features like change of operations mode, change of the accelerator and many more are supported by extensive use of the observer pattern [16].

The framework requires only the standard Java libraries. This eases a consistent deployment, since no library can be forgotten by the developer, leading to strange runtime errors. Elements from other libraries - like the Spider - will only be shown, if the hosting library is supplied by the developer (creation by reflection).

CONCLUSION

MstApp improves the operator experience, application developer productivity and delivered software quality. A

common look and feel is achieved and at the same time application developers are relieved from many mundane tasks. Common tasks are only written and tested once. Many changes to the environment can be adapted to in one place only.

ACKNOWLEDGEMENTS

The framework MstApp - as a simple central hook into the control system - has many more authors than this paper. There are major contributions from Reinhard Bacher, Piotr Bartkiewicz, Andreas Labudda, Marcus Walla and Franziska Wedtstein. It is a pleasure for us to thank all mentioned and unmentioned contributors.

REFERENCES

[1] Reinhard Bacher, "Commissioning of the New Control System for the PETRA 3 Accelerator Complex at Desy", Proceedings of ICALEPCS 2009, Kobe, Japan.
[2] Piotr Bartkiewicz, DESY, Hamburg, private communication
[3] Marcus Walla, DESY, Hamburg, private communication
[4] http://pub.cosylab.com/acop/site/AcopSpider.html
[5] Andreas Labudda, "Building and Deploying loosely coupled Console Applications", Proceedings of PCaPAC 2006, Newport News, USA, p 126
[6] Jürgen Maaß, Georg Mann, "PETRA III Operation Modes", DESY MCS1 internal paper 2005-02-04
[7] http://java.sun.com/products/jfc/tsc/articles/persistence4/
[8] Josef Wilgen, DESY, Hamburg, private communication
[9] Rüdiger Schmitz, "What's Behind an Accelerator-Control System?", Proceedings of PCaPAC 2010, Saskatoon, SK., Canada.
[10] Philip Duval, http://tine.desy.de/
[11] R. Kammering et al., "E-logbook Reloaded - or the Renovation of DESYs Electronic Logbook", Proceedings of ICALEPCS 2009, Kobe, Japan.
[12] Philip Duval, http://adweb.desy.de/mcs/tine/TineSecureServices.html
[13] Philip Duval, TINE Release 4.0 News, April 18, 2008; http://adweb.desy.de/mcs/TINE_Users_Meeting/ 2002Apr18/Release4News.pdf
[14] http://doocs.desy.de/
[15] http://jddd.desy.de/
[16] E. Gamma et al., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional, 1994

DATA ANALYSIS WORKBENCH

A. Götz , M.Gerring, O.Svensson, ESRF, Grenoble, France
S. Brockhauser, EMBL, Grenoble, France.

Abstract

Data Analysis Workbench is a new software tool being developed at the ESRF. Its goal is to provide a tool for both online data analysis which can be used on the beamlines and for offline data analysis which users can use during experiments or take home. The tool includes support for data visualization and workflows. Workflows allow algorithms which exploit parallel architectures to be designed from existing high level modules for data analysis in combination with data collection. The workbench uses Passerelle as the workflow engine and EDNA plugins for data analysis. Actors talking to Tango are used for sending commands to a limited set of hardware to start existing data collection algorithms. A Tango server allows workflows to be executed from existing applications. There are scripting interfaces to Python, Javascript and SPEC. The current state at the ESRF is the workbench is in test on a selected number of beamlines.

INTRODUCTION

The Data Analysis Workbench (DAWB) project has been started to create a platform for integrating data analysis programs. Data analysis programs usually exist as standalone programs. Each program has its own user interface, graphical and/or textual. Each program has its own internal architecture and choice of language and libraries. This makes it extremely difficult if not impossible to share code between programs.

Users are confronted by a myriad of different programs and interfaces to manage. Users generally revert to scripting languages to get around this problem. Scripting languages like Python allow them to provide a level of automation for running analysis programs. However scripts are usually not destined to be shared, documented or distributed. Scripts are also not well adapted to visualisation.

DAWB (see Figure 1) adds a layer on top of scripts to make data analysis and visualisation accessible to everyone and repeatable by introducing the concepts of a workbench, visualisation, scripting, workflows and experiment and data analysis design.

CONCEPTS

The goal of the Data Analysis Workbench is to provide powerful data analysis tools, both for online (synchrotron radiation beamlines) and offline data analysis. The combination in the workbench of a workflow tool (Passerelle, based on Ptolemy II) and a data analysis framework (EDNA) makes it straight-forward to develop advanced but robust data analysis workflows. Prototyping of workflows is facilitated by the workbench data visualisation support which makes it easy to visualise 1D and 2D data. The TANGO framework is used for online data analysis for both integrating beamlines hardware and for remotely controlling the workflow engine from the beamline GUI.

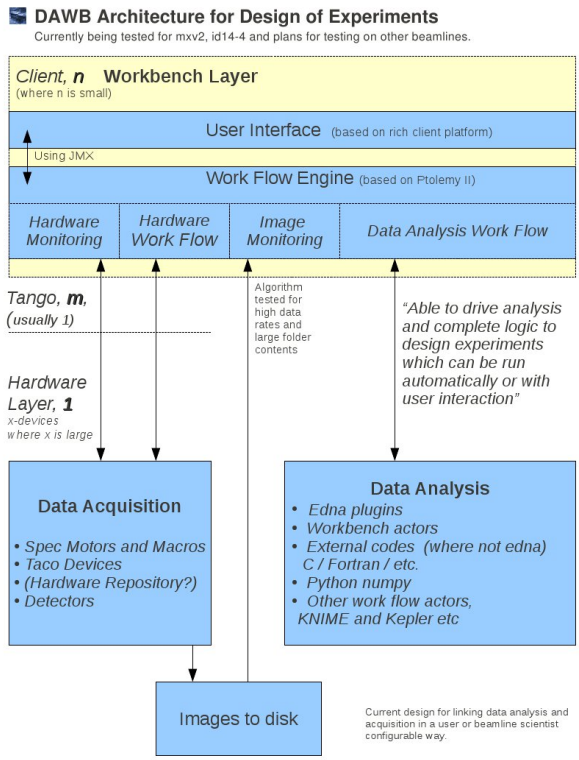


Figure 1: DAWB Architecture.

To achieve these goals the data analysis workbench is based on the following concepts:

Workbench

Data analysis is a sufficiently complex task that a simple approach of a main window with a few buttons is not sufficient. The Eclipse Workbench has been adopted as base platform for the data analysis workbench. The workbench can be best described by quoting the Eclipse documentation*: "*The Workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources. Each Workbench window contains one or more*

* http://help.eclipse.org/indigo/index.jsp

perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars. More than one Workbench window can exist on the desktop at any given time."

Visualisation

Data analysis is a means to an end. In the initial, intermediate and final steps the raw and analysed data need to be visualised. One of the workbench's main goals is to offer a complete set of visualisation tools in 1D, 2D, 3D and the possibility of following changes as a function of time. Currently 1D and 2D are implemented. The aim is to offer viewers for all types of files typically used in data analysis in a single tool. Users need to learn one set of viewers for all data analysis applications called from the workbench.

Scripting

Scripting is an essential part of data analysis. It allows existing codes to be glued together rapidly and in a flexible manner. The workbench provides support for Python and Javascript as scripting languages. There is basic support for SPEC. Others languages might be added in the future. Python support includes a complete language development environment – PyDev[†]. Scripts can be called as standalone from a terminal or as part of a workflow.

Workflows

Workflows are a convenient way of programming and documenting the flow of data through algorithms. They can also be used to feedback to data acquisition systems when designing experiments. Workflows are naturally parallel and ease the programming of multiple execution threads. DAWB provides strong support for workflows, both online and offline. They are seen as a complement of scripts. Workflows are higher level constructs which assist the beamline scientist and user in building robust pipelines which use scripts.

Features

Import files as links - import large data directories into a project structure as links. This reduces file system overheads.

Open files in various formats - srs, dat, png, jpg, jpeg, tif, tiff, cbf, img, ciff, mccd, edf, pgm, cor, bruker, h5, nxs, pdb, (gz, bz2, zip)

Visualization - able to visualize data in 1D and 2D.

Workflow editing - viewing and editing of workflow data analysis pipelines based on Ptolemy 2.

Running of workflow - able to run workflows in a separate process which is controllable from the GUI and able to run without the user interface.

Eclipse projects - able to keep data in eclipse projects and analyze data further using user interface tools and python.

Drag and Drop - support of drag and drop of data from eclipse projects to Passerelle workflows.

Tango Devices - the monitoring of tango devices and the interaction with them via actors in the workflow.

Image Monitoring - the ability to monitor a directory of images using a thumbnail viewer, even if the directory contents are large and changing quickly.

HDF5 / Nexus - read and write hdf5 including slicing. Plotting of 1D and image datasets and slicing higher dimensions. Supported on a wide range of platforms and in multi-threaded environments. Nexus files in hdf5 also supported.

Data analysis - the support for python actors and EDNA actors natively and connections via system process actors for Fortran, C/C++ and other executable programs. EDNA is the most heavily used framework for integrating arbitrary data analysis to the workbench at the moment.

ECLIPSE/RCP

Eclipse/RCP[‡] is an essential part of DAWB. Eclipse/RCP offers a robust platform for building rich clients. It has many features which are useful for managing a workbench. The fact that it is a commercial quality open source product makes it easy to collaborate with commercial and non-commercial partners. The high quality documentation and large number of resources on the web make it easy to learn. Eclipse/RCP has enabled DAWB to be developed in a way that other institutes who have adopted Eclipse/RCP can use it even if they were not involved in the definition of or aware of DAWB.

WORKFLOWS

Workflows are relatively new in the synchrotron data analysis field. They are however widely used in other scientific fields like biology. They offer a higher level programming language than traditional textual languages e.g. Python, C and Fortran. Their goal is not to replace these languages but to complement them by providing a higher level tool for calling programs in these languages. For this reason workflows should remain high level and not become too fine grained. Too many nodes or graphs in a workflow quickly make it difficult to follow or maintain. Workflows like Passarelle in DAWB should not be confused with programming languages like Labview's graphical programming language G. In the case of the latter the goal is to do as much programming as possible in this language. It is not easy to interface it to scripting

[†] http://pydev.org/

[‡] http://www.eclipse.org/home/categories/rcp.php

languages like C or command line driven programs. DAWB on the other hand encourages users to delegate tasks as much as possible to high level modules (which could be scripts or standalone programs). The workflow is then dedicated to documenting the data flow and adding parallelism to the process. Workflows with the help of appropriate actors make accessing remote sources like web services, batch schedulers and cloud services transparent.

Workflow TANGO Server

The workbench is very good for developing workflows and running them when visualisation is part of the workflow. For some applications however once the workflow has been developed the workbench is not needed anymore. In those cases it is preferable to run the workflow "headless" i.e. without the graphical user interface. A TANGO device server has been written for this purpose. It implements start and abort commands plus the possibility of passing parameters to and from the workflow during its execution. It implements a state machine which reflects the status of the workflow's execution. The device server facilitates the integration of workflows in existing applications like MxCube [4].

MX workflow examples

Experiments performed at macro-molecular crystallography (MX) beamlines can to a large extent be automated [4]. However, the automation of data acquisition (beam delivery, sample changer, centring, mxCuBE) and data analysis (EDNA characterisation [5] and automatic data processing) have up until now been largely decoupled with only few possibilities of data analysis to influence data acquisition. New highly automated beamlines like MASSIF (ESRF upgrade program) will take advantage of advances in beam delivery and automation of sample handling. Together with recent advances in data analysis software packages this creates higher demands in term of overall automation of both data acquisition and data analysis.

The ESRF MX beamlines have successfully used DAWB for developing four workflows which significantly increases the coupling between data analysis and data processing:

Enhanced EDNA characterisation workflow for characterisation of MX crystals. This workflow can automatically re-adjust exposure time, oscillation width and detector distance in case these were not set optimally before collecting the reference images.

Crystal radiation sensitivity measurement workflow for precisely measuring the radiation damage susceptibility of crystals by sacrificing a crystal (or a part of a crystal).

Kappa goniometer re-orientation workflow (see Figure 2) for re-orienting a crystal to the desired orientation, e.g. aligning the principal unit axis along the spindle axis or aligning an even-fold symmetry axis along the spindle axis.

Mesh scan workflow for finding the part of a crystal which gives the best data by scanning in two dimensions a low-intensity beam over the crystal and at the same time evaluate the diffraction intensity.

A paper which describes these workflows in more detail has been submitted [3]. The workflows are planned to go into production later this year.

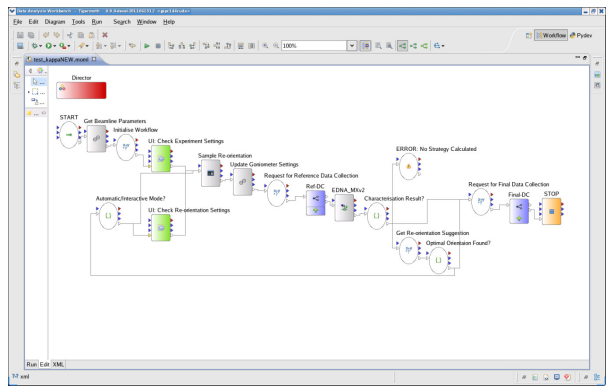


Figure 2: Workbench showing Kappa reorientation workflow for automatic characterisation of MX crystals

EXAFS workflow example

JESF is a Fortran code written for spectroscopy experiments which has been used for obtaining fast, first pass, data analysis at the ESRF. This code can be run online or with "zero dead time" attached to the detector using DAWB. At the ESRF, this works with shared memory. Since JESF uses a file and the speed requirements of the pipeline (2Hz maximum) are not currently large, we decided to write the data to a file and then run JESF on the file. However DAWB allows in memory operation as well. After JESF is run, the result files are opened in real time, so the plots refresh during the run. This pipeline has been tested with a SPEC macro writing to the shared memory currently as the beamline is not yet commissioned. DAWB allows this pipeline to be constructed / modified quickly and delivered to the beamline in a robust way. See Figure 3 for a screenshot of the JESF workflow.

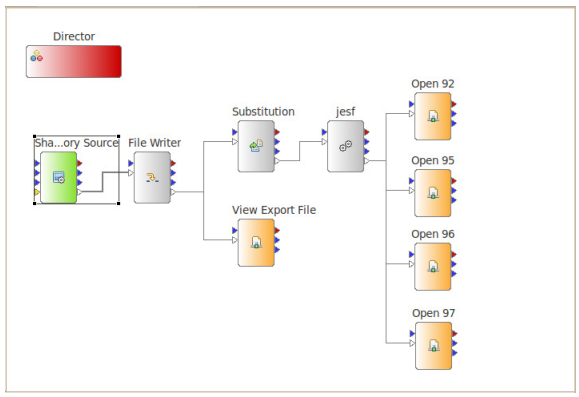


Figure 3: JESF Workflow for EXAFS online data analysis

Future Workflows

The capability of flexibly combining online-data-analysis actors with beamline control actors as well as reusing complete workflows as composite actors in a more complex workflow enables rapid implementations of new experiment protocols. These can include full 3D sample quality characterisation via diffraction tomography, evaluating multiple crystals in a raw sample to find the best diffracting crystal or even the best place and orientation of the best quality crystal. Using the already mentioned couplings of EDNA actors with the Data Collection composite actor, like Radiation Sensitivity Workflow, or Enhanced Characterisation Workflow, even more sophisticated workflows can be built e.g. chaining them one can gain empirically a susceptibility coefficient in the first step and then immediately use it during the characterisation for more precise results. Chaining basic calibration workflows one can automate complete beamline checking protocols. The list of possible workflow programming is endless and comparable to the list of scripts. To gain the most out of workflow design, it is important to keep the structure of the workflows clean and easy-to-understand (comments on the canvas if needed and documentation of course) as well as defining a proper data model to be used all along the communications between the actors, so that the dataflow is kept clean.

COLLABORATION

DAWB is an Open Source program and welcomes collaborators. The source code is on a publicly accessible repository[§]. The source code is released under a modified GPL[**] licence which allows actors developed by third parties not to be under GPL. Workflows developed with DAWB are not bound by the DAWB licence. They can be released under the licence of choice of the developers. Open source licences are encouraged because they allow other developers to study the source code and improve it and or learn from it.

CONCLUSION

DAWB has demonstrated that it is possible to provide a commercial quality workbench for synchrotron data analysis which supports workflows, visualisation and scripting for online and offline data analysis. The first examples of workflows running on MX and EXAFS beamlines are very promising. We think that in the future workflows are going to be an essential part of managing high-level scientific data analysis. DAWB has the features necessary to achieve this. We encourage others who are interested in scientific workflows either as developers or as users to give DAWB a try[††] and if interested to join the DAWB collaboration. We think we should collaborate on the core tool but leave the choice to compete or collaborate on workflows open.

ACKNOWLEDGEMENTS

We would like to acknowledge the following groups: iSencia[‡‡] for providing the Passerelle workflow tool under an open source licence and for doing the Eclipse/RCP version, Soleil for introducing us to Passerelle, Diamond for making parts of GDA and SDA available under an open source licence.

REFERENCES

[1] Johan Eker et. al (2003). “Taming Heterogeneity - The Ptolemy Approach” in Proceedings of IEEE, vol. 91, no. 1

[2] Gwenaëlle Abeillé, Majid Ounsy, Alain Buteau, “A Graphical Sequencer for SOLEIL Beamline Acquisitions”, ICALEPCS 2007 Proceedings

[3] Sandor Brockhauser et. al. “The use of Workflows in the design and implementation of Complex Experiments in Macromolecular Crystallography””, to be published in Acta D

[4] Antonia Beteva et. al. (2006). “High-throughput sample handling and data collection at synchrotrons: embedding the ESRF into the high-throughput gene-to-structure pipeline”, Acta Cryst. D 62, 1162-1169.

[5] Marie-Francoise Incardona, et. al. (2009). “EDNA: a framework for plugin-based applications applied to X-ray experiment online data analysis”, J. Synch. Rad. 16, 872-87

[§] http://code.google.com/a/eclipselabs.org/p/dawb/

[**] http://www.gnu.org/copyleft/gpl.html

[††] http://dawb.org

[‡‡] http://www.isencia.com/

ADDING FLEXIBLE SUBSCRIPTION OPTIONS TO EPICS*

Ralph Lange, Helmholtz-Zentrum Berlin / BESSY II, 12489 Berlin, Germany
Andrew Johnson, Argonne National Laboratory, Argonne, IL 60439, USA
Leo Dalesio, Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

The need for a mechanism to control and filter subscriptions to control system variables by the client was described in a paper at the ICALEPCS2009 conference [1]. The implementation follows a plug-in design that allows the insertion of plug-in instances into the event stream on the server side. The client can instantiate and configure these plug-ins when opening a subscription, by adding modifiers to the channel name using JSON notation [2]. This paper describes the design and implementation of a modular server-side plug-in framework for Channel Access, and shows examples for plug-ins as well as their use within an EPICS control system.

MOTIVATION

Over the years, more and more powerful front end computers have lead to increasing data processing rates. Event and timing systems nowadays allow to attach accurate synchronous nanosecond resolution time stamps to data.

Many clients will not need every data update available. Those clients will either want to reduce the update rate for their subscription, or have the updates being correlated to events from the event system, so that they are only getting updates during phases they are specifically interested in.

The EPICS (Experimental Physics and Industrial Control System) toolkit [3] and its network protocol, Channel Access [4], only provide a small set of server-configurable update rates, and only very limited correlation between updates and a timing/event system.

SERVER-SIDE PLUG-IN FRAMEWORK

Considerations

When designing an extension to a widely-used, highly scalable system, a number of considerations have to be taken into account:

- Network compatibility: changing the on-the-wire protocol of Channel Access should be avoided.
- Footprint: the extension should be modular and only add minimal resource use on embedded systems.
- API compatibility: internal APIs should be left intact to avoid breaking externally developed code.

Existing Update Mechanism

The existing update mechanism of the EPICS database has been described in detail in [1] and is shown in Fig. 1. At connection time, a set of per-client tasks and event queues is set up. When the database processes an EPICS record, events are generated from the subscription definitions linked to the record, and put into the event queue. The Channel Access event tasks take events off the queue and ship it to the network.

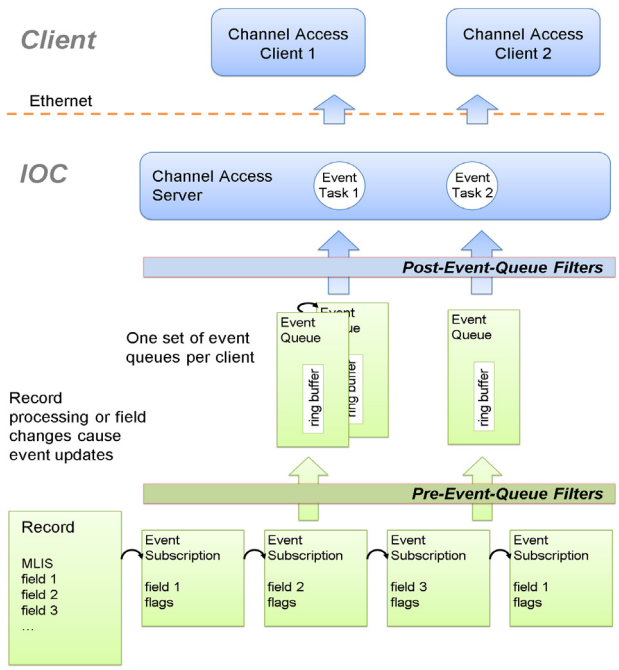


Figure 1: Event Update Mechanism.

Plug-In Framework

The required additional processing of update events is done in plug-ins, that a Channel Access client can request to be inserted into the event stream when the connection is made. These plug-ins have an API that allows for receiving and generating event data, so they can be stacked.

When update events are moving through the plug-in stack, they are handed over to each of the plug-ins. This allows the plug-in to change the event's data as well as the meta data (type, size, time stamp, alarm information etc.) The plug-in may also drop or create additional update events.

Instantiation and configuration of plug-ins is controlled by the client through JSON (JavaScript Object Notation) channel name modifiers, that are transparently forwarded through Channel Access to the IOC [2]. A JSON parser, that has been added to the IOC core software recently, is used for parsing the modifier.

*Work supported by U.S. Department of Energy (under contracts DE-AC02-06CH11357 and DE-AC02-98CH10886), German Bundesministerium für Bildung und Forschung and Land Berlin.

The plug-in framework offers two levels for integration of plug-ins:

A *channel filter* directly implements callbacks for the JSON parser, and thus can use any transmitted JSON structure as configuration.

A *channel filter plug-in* uses a simpler convenience API provided by the framework. Configuration data is restricted to key-value pairs of basic types, but the framework does the parsing for the plug-in, and the simple API allows for easy, efficient, and safe addition of plug-in modules.

The event-related part of both APIs is equivalent.

Adding Plug-Ins to an IOC

All plug-ins have to register themselves with the framework before use, specifying their name (that clients will use for instantiation) and a pointer to their API implementation. The plug-in registration can take place at any time, even after IOC initialization. That would allow plug-in code to be loaded and registered on-demand in a running IOC.

Two Target Layers

The first part of the update mechanism, between record and the event queues (shown *green* in Fig. 1), is done as part of the EPICS database processing, i.e. at high priority. CPU use in this context may keep other records from processing and affect the IOC's real time behavior.

The second part of event generation, between the event queue and the network (shown *blue* in Fig. 1), is done as part of the Channel Access tasks, i.e. at low priority. Database processing will not be affected by CPU usage in this context.

Some plug-in functionality needs to be instantiated on the database side. Plug-ins that reduce the update rate by averaging or limiting should drop surplus events before putting them on the event queue. Plug-ins that correlate updates with an external event system have to do that within the context of record processing.

Other plug-in functionality should be executed in the low priority networking context. Plug-ins that manipulate the data of a single update without relation to other updates, e.g. extracting parts of an array, doing CPU intensive mathematical operations, should avoid affecting the real time database operation, and only do these expensive operations on updates that are actually shipped to the client.

The framework offers instantiation in both layers. Fig. 1 shows the location of the two plug-in layers in the existing update scheme.

PLUG-IN COLLECTION

A number of plug-ins of different complexity have been implemented, to test function and validity of the framework APIs.

ts – Timestamp "Now"

This pre-event-queue plug-in sets the time stamp of the update to "now" (the current time).

Usage: myPV.A{"ts":{}}

When subscribing to a field that does not cause record processing, the data updates are stamped with the time stamp of the record's last processing, which does not show the time the field was updated. This plug-in solves the issue.

dbnd – Deadband Throttling

This pre-event-queue plug-in implements deadband based update throttling, similar to the mechanism in the EPICS records.

Usage: myPV.RVAL{"dbnd":{"m":"rel","d":7.5}}

As a plug-in it is available to any record field, offers per-client configuration, and adds relative deadband specification.

arr – Array Subset

This plug-in inserts itself in the post-event-queue layer. It creates a sub-array with the specified start, increment, and stop index.

Usage: myArray.VAL{"arr"{"s":-5,"i":2}}
myArray.VAL[-5:2:]

The second form is a shorthand notation added for convenience.

sync – Synchronize with Timing System

This pre-event-queue plug-in synchronizes data updates with internal or external timing systems by pushing updates only under certain conditions.

Usage: myPV.VAL{"sync":{"m":"while","s":"red"}}

Synchronization is done with respect to *system state*. This is implemented by means of a small library that offers an API through which named states can be defined and set or reset. Figure 2 shows the effect of the six available synchronization options with respect to a state:

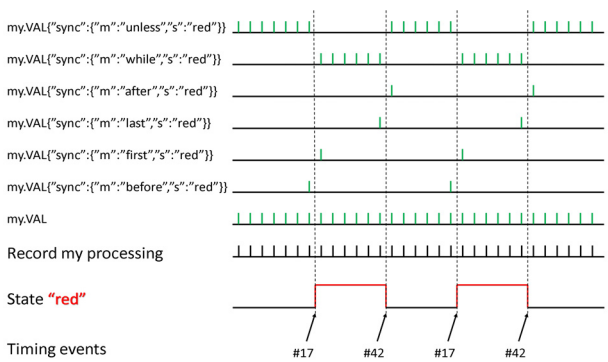


Figure 2: Options of sync Plug-In.

- *unless*: All updates while state is false are being sent.
- *while*: All updates while state is true are being sent.

- *after*: Only the first update after the transition of state from true to false is being sent.
- *last*: Only the last update before the transition of state from true to false is being sent.
- *first*: Only the first update after the transition of state from false to true is being sent.
- *before*: Only the last update before the transition of state from false to true is being sent.

IMPLEMENTATION EXPERIENCES

The major part of the framework implementation was related to reorganization of the subscription definition and data update structures. The subscription structure changes allow per-plug-in and per-plug-in-instance configuration data to be linked to a subscription. The data updates were changed from statical allocation in fixed size lists inside the event queue to heap-allocated structures that use free lists to minimize memory allocation and deallocation at run time.

While implementing the plug-ins described above, the channel filter plug-in API was carefully extended and optimized. In the current state, the individual plug-ins use very little code (40-150 lines of C), and mainly implement the code for their core functionality, i.e. their data update modifications. Thus, implementation of the set of plug-ins described above was fast and uncomplicated.

The two target layer design has proven to be the right approach, as the plug-in designer can freely select the appropriate context for a plug-in.

FOOTPRINT

The framework itself does not add much to the EPICS record structures and code. When no plug-ins are used, the additional memory and computing power consumption is negligible.

Small and embedded EPICS systems will not run into problems because of the plug-in framework.

STATUS

The framework has been developed based on EPICS base version 3.14. The code base is complete and working, unit test code has been written to verify most of the functionality.

Next steps are merging the code into the version 3.15 development trunk, combined with a careful review of the framework's memory allocation techniques and error management – two areas where standards have changed for EPICS base version 3.15.

FURTHER PLANS

Additional plug-ins will allow to throttle data updates by setting the maximum update rate, execute atomic put/get operations, and perform statistical operations on array data.

Records will be able to set default configuration values for certain plug-ins through info tags in the EPICS database.

CONCLUSION

The addition of the server-side plug-in framework to the EPICS toolkit adds valuable functionality to the IOC. It opens yet another way to customize the system, allowing to add exactly the operations needed for a specific installation, and configure them at run time individually for the appropriate connections.

This framework helps to overcome limitations of the existing design, and allows the future addition of new functionality without creating a major impact on performance, memory footprint, or overall complexity.

REFERENCES

[1] R. Lange, A.N. Johnson, “Advanced Monitor/Subscription Mechanisms for EPICS”, THP090, Proceedings of ICALEPCS2009, Kobe, Japan, pp. 847-849.

[2] A.N. Johnson, R. Lange, “Evolutionary Plans for EPICS Version 3”, WEA003, Proceedings of ICALEPCS2009, Kobe, Japan, pp. 364-366.

[3] Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics.

[4] J. Hill, R. Lange, “EPICS R3.14 Channel Access Reference Manual”, http://www.aps.anl.gov/epics/base/R3-14/12-docs/CAref.html.

EPICS V4 IN PYTHON*

Guobao Shen#, Marty Kraimer, Michael Davidsaver, BNL, Upton, NY 11973, U.S.A.

Abstract

Interest in Python as a rapid application development environment continues to grow. Many large experimental scientific facilities have adopted Python for beam commissioning and the operation. The EPICS control system framework has become the de facto standard for the control of large experimental facilities, where it is in use in over 100 facilities. The next version of EPICS (EPICS V4), under active development will extend the support for physics applications, data acquisition, and data analysis. Python support for EPICS V4 will provide an effective framework to address these requirements. This paper presents design, development and status of activities focused on EPICS V4 in Python.

MOTIVATION

As part of the EPICS V4 [1] initiative, there is increased interest in expanding the conventional low level hardware support to include high level applications such as physics applications, data acquisition and data analysis [2]. The support consists of the following main modules: (1) pvData, a memory resident real time database with a predefined data structure; (2) pvAccess, a network protocol for transferring data over wire; (3) pvIOC, a processing engine; (4) pvService, a collection of pvIOC instances implemented as a service.

The high performance/low latency of EPICS V3 for stream data and instrumentation control has been thoroughly demonstrated in many facilities. To implement the proposed extensions to EPICS V4, especially data acquisition, a major concern is its performance. Benchmark tests have been conducted at NSLS-II project to compare the performance of pvAccess with that of Channel Access [3][4]. These benchmarks were performed using pvAccess implemented in Java, and Channel Access implemented in C/C++. The results show that for scalar data, pvAccess has identical performance when processing large numbers channels, and a slightly reduced performance when processing only a few (for example 10) channels. For array processing, pvAccess outperformed Channel Access. More detailed benchmarking results are given in [3]. The benchmark results satisfied the requirements to proceed with the proposed enhancements.

A modular infrastructure based on the client/server model has been designed for NSLS-II for beam commissioning, physics study, and beam operation applications [5][6][7]. In this approach, data flow is separated to 2 parts: (1) acquire data from hardware, and (2) consume data in the application. The 2 flows are implemented with a well-defined API. An advantage of this design is that hardware data flow API is implemented by developers, who have more experience on data and hardware control. The data consumption is handled by physicists who are experts in data analysis.

The client/server implementation is based on EPICS V4, which provides a good technical framework for the infrastructure, the flexibility of data structure design, and satisfactory performance for the physics applications. The 3-tier architecture is shown in Fig. 1. The detailed explanation of this architecture can be found in [8][9].

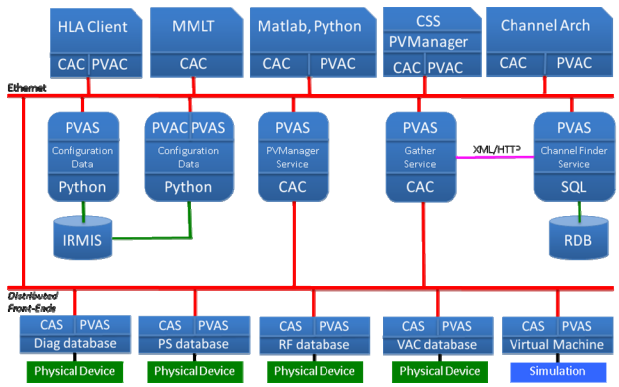


Figure 1: System Architecture.

Figure 1 illustrates the application development environment provided by EPICS V4. Developers have the flexibility to select their favourite technologies to develop back end applications.

At NSLS-II, Python has been selected as the primary development language for physics applications. This motivated us to provide client side Python support. In addition, as shown in Fig. 1, all static data stored in the IRMIS database are served through the V4 server layer. A Python API is under development for access to the IRMIS database. To utilize this API, it is required also to have a Python support on server side.

These 2 factors provided the motivation to have full Python support for EPICS V4. In this paper, we report the results and present current development status for Python support using 3 separate approaches.

PYTHON IMPLEMENTATION

As an interpreted, high-level programming language with dynamic semantics and native support of extensive scientific libraries, Python is very attractive for rapid application development as well as for use as a scripting or glue language. There is a growing interest in the community to use Python as the physics application development platform. At the NSLS-II project, Python has been selected as primary language for physics application development.

EPICS V4 was developed originally in Java, and provides a comprehensive set of features. Later as

*Work supported under auspices of the U.S. Department of Energy under Contract No. DE-AC02-98CH10886 with Brookhaven Science Associates, LLC, and in part by the DOE Contract DE-AC02-76SF00515

#shengb@bnl.gov

requested by the community, a C++ binding was developed and now has full functionality in both the pvData and pvAccess modules, and basic support in the pvIOC and pvService modules. The current language bindings are shown as Table 1.

Table 1: EPICS V4 Language Bindings.

Module	Java	C++	Python
pvData	Y	Y	Y/Limited
pvAccess	Y	Y	Y/Limited
pvIOC	Y	Y/Limited	N
pvService	Y	Y/Limited	N

In this paper, we will focus on the development in Python. For the Python binding, there are several possible solutions: (1) utilizing Python/C API; (2) using existing wrapping tools; and (3) developing a native binding all in Python.

In this section, we will discuss the first two, followed by a discussion of the native binding development the next section.

Python/C

Python/C provides an API for a programmer to build an external module callable in Python. It is a relatively well-understood process to writing an extension module. Most Python/C API functions have one or more arguments as well as a return value of type PyObject*. The returned type is a pointer to an opaque data type, which represents an arbitrary Python object. All Python object types are generally treated the same way in the Python language.

We have to pay attention that all Python objects have a type and a reference count. When exposing an API to Python, the developer must explicitly manage the reference count in the C code. When an object's reference count becomes zero, the object is de-allocated, and collected by garbage collector.

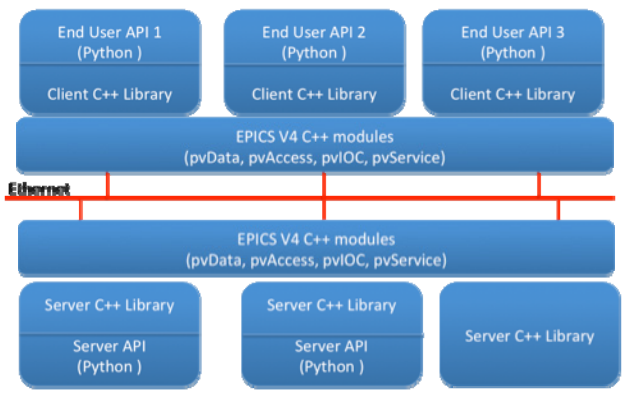


Figure 2: Software Structure Using Python/C.

A realistic approach, shown in Fig. 2, for the end user client interface is to develop a client library in C++, and wrap the C++ library as a Python interface. Each client library is standalone. The benefits are that most codes are in C++, and in general we can gain performance in Python as good as that in C++. However, changes in EPICS V4 C++ modules usually affect the client C++ library. As the project grows, the number of end user client libraries will increase. Long-term maintenance will become an issue.

Wrapping Tools

To minimize the effect to the client library caused by changes of V4 C++ modules, another approach is to wrap each C++ module. This software architecture is as shown in Fig. 3. The end user API is developed in native Python. Any change in the EPICS V4 library affects only the wrapper library, and is transparent to the end API developer/user. Another advantage with this solution is that both client and server can use the same wrapping library.

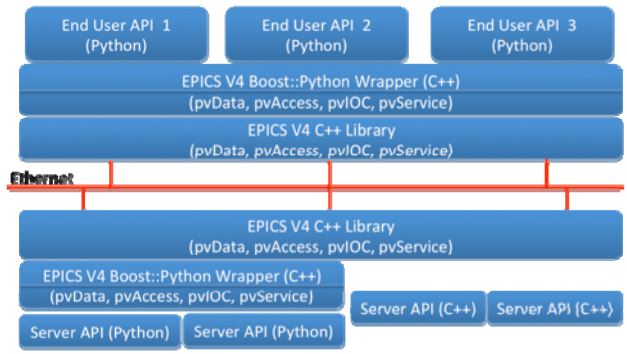


Figure 3: Software Structure by Wrapping V4 Modules.

Tools such as SWIG, SIP, and Boost Python [10] can be used to expose C++ API to Python. Essentially, they are a wrapper for the Python/C API, and provide a more realistic solution to solve many well-known problems. For example, using the Python/C API, the developer has to deal with pointers passed between Python and C++. The developer is responsible for pointer management, particularly when the referenced object has been deleted. Using wrapping tools, those problems are taken care by the tools.

At NSLS II, 2 tools are evaluated, SIP and Boost Python respectively. Because SIP does not support the STD::TR1 smart pointer well, Boost Python was selected in the end. The Boost Python is one member of the Boost C++ library collection, and binds C++ and Python in a mostly-seamless fashion. It also provides a set of policies to track the ownership of an object in both C++ and Python domains. With the Boost Python Library, the developer can quickly and easily export C++ to Python.

A disadvantage with Boost Python library is that a boost library must be installed, and care has to be taken to switch between the boost library and the STD library. Moreover, the Boost Python library does not fully support STD::TR1 library, and provides a simulated interface set for STD::TR1.

Native Implementation

Either the Python/C approach or exposing C++ API to Python using wrapping tools is a workable solution.

Although exposing the C++ API allows users to focus their development on native Python, this heavily relies on the Boost Python Library. This library dependency forces all of EPICS V4 to be compiled against the Boost library. Moreover, tracking and passing object ownership to/from between C++ and Python domain is difficult.

Although we believe Boost Python is best approach for exposing the EPICS V4 API to Python, its template meta-programming mechanism pushes compilers to their limits. Often it is necessary to increase template-depth limit settings of the compiler, memory usage can be vast, compile times are long, and error messages are often difficult to decipher.

A new design is under consideration to implement EPICS V4 in native Python to take full advantage of Python. For example EPICS V4 defined an efficient way to describe complex data structures and the data protocol. In the current implementation in either C++ or Java new data structures must be constructed entirely from the primitive data types. Using Python, it is much simpler to map the data structure into a numpy [11] array. The software architecture is shown in Fig. 4.

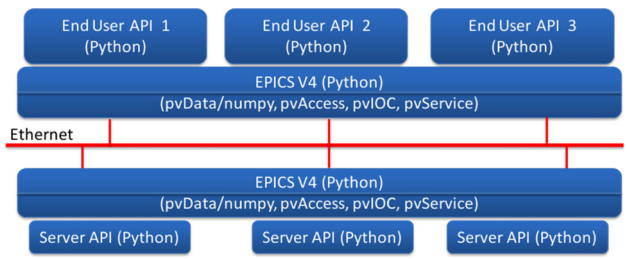


Figure 4: Software Structure in Native Python.

CURRENT STATUS

Python/C

As above described, each library developed in Python/C is a standalone library. In another words, its development relies on each service development. Currently, one library is developed for a service implemented under EPICS V4 in Java, a 'gather' service. The library supports all functions required by clients, and supports all fundamental data manipulation such as get, put, and monitor.

Boost Python Wrapping

With Boost Python library, basic APIs have been implemented in Python for the pvData and pvAccess modules. The end user interface for accessing the gather service is under development. Design and implementation issues related to thread safety and locking have been found and resolved in the C++ modules.

Basic unit tests have been performed for the exposed APIs. More detailed testing is necessary to improve the stability and robustness.

Native Implementation

This is still under design, and the development is at very early stage. Some preliminary code has been developed to verify basic ideas, for example mapping EPICS V4 data structures into numpy arrays.

SUMMARY

This paper presented activities on Python development for EPICS V4. Several different approaches have been conducted at BNL and the status is described.

At the present stage, a workable approach is to utilize Boost Python library. In this approach, there is a trade off solution between performance and flexibility/scripting. Performance benchmarking is necessary.

ACKNOWLEDGEMENT

The authors would like to thank Matej Sekoranja at COSYLAB for his contributions on epics-pvdata development, especially the pvAccess implementations in both Java and C++. They want to give their thanks to Leo Bob Dalesio at BNL for his continuous support and encouragement.

REFERENCES

[1] http://epics-pvdata.sourceforge.net/
[2] L. Dalesio, *et al*, "EPICS V4 Expands Support to Physics Application, Data Acquisition, and Data Analysis", This proceedings, FRBHMULT06, Grenoble, France (2011)
[3] G. Shen, "Performance Analysis of EPICS Channel Access and pvAccess", NSLS-II Tech Note 082 (2010)
[4] G. Shen, *et al*, "Server Development for NSLS-II Physics Applications and Performance Analysis", Proc. of PAC11 (2011), MOP252, New York, USA
[5] G. Shen, "A Software Architecture for High Level Applications", Proc. of PAC09 (2009), FR5REP004, Vancouver Canada
[6] G. Shen, "A Modular Environment for High Level Applications", Proc. of ICALEPCS09 (2009), THP094, Kobe Japan
[7] G. Shen, *et al*, "A Novel Approach for Beam Commissioning Software using Service Oriented Architecture", Proc. of PCaPAC10 (2010), WEPL037, Saskatoon Canada
[8] G. Shen, *et al*, "Prototype of Beam Commissioning Environment and its Applications for NSLS-II", Proc. of IPAC10 (2010), WEPEB026, Kyoto Japan
[9] G. Shen, *et al*, "NSLS-II High Level Application Infrastructure and Client API Design", Proc. of PAC11 (2011), MOP250, New York, USA
[10] http://www.swig.org/; http://riverbankcomputing.co.uk/software/sip/intro; http://www.boost.org/
[11] http://numpy.scipy.org/

MANGO: AN ONLINE GUI DEVELOPMENT TOOL FOR THE TANGO CONTROL SYSTEM*

G. Strangolino, C. Scafuri, Sincrotrone Trieste, Trieste, Italy

Abstract

Mango is an online tool based on *QTango* [1] that allows easy development of graphical panels ready to run without need to be compiled. Developing with Mango is easy and fast because widgets are dragged from a widget catalogue and dropped into the Mango container. Widgets are then connected to the control system variables by choosing them from a *Tango* [2] device list or by dragging them from any other running application built with the *QTango* library. Mango has also been successfully used during the FERMI@Elettra commissioning both by machine physicists and technicians.

WHAT MANGO IS

Mango is a simple tool to generate online *QTango* panels, that are ready to run without need to be compiled. In this sense, Mango is an interpreter. The interfaces generated with Mango are made up of the simple *QTango* and *Qt4 [3]* widgets, and all the logic of the components and their interaction with each other relies on the intrinsic logic of the widgets themselves.

Whom Mango is Addressed to

- The *Tango* device server programmer, who wants to design a Tango device fully compatible with the *QTango* library and wants to immediately test his work in progress. He might even outline a GUI choosing the most suitable widgets to represent the Tango attributes and commands that are modelled by the device server. The sketch can be saved and submitted to the attention of the GUI developer, who can complete it, improve some human computer interaction aspects and even import the sketch into the *Qt4 designer* to create an executable.
- The GUI developer who wants to rapidly create an interface to a *Tango* device server whose commands and attributes fully support the *QTango* widgets, without needing any additional logic.
- The *end user* of the graphical interface, who can outline a draw of the desired panel and discuss it with GUI developer.
- The control room operator. He might use Mango to create *on the fly* panels, in order to create a summarizing graphical interface gathering *QTango* objects from other real *QTango* executable applications running on the desktop. Actually, the powerful *drag and drop* system exported by the *QTango* library allows dragging a *QTango* widget from an application and dropping it into any Mango container.
- The *QTango* library tester: Mango widgets interacting with the Tango framework are *derived* from the corresponding *QTango* elements and represent valid test objects, both from the stability point of view and from the good design of the elements (the better the widgets are designed, the better possibilities are offered to the Mango tool to manage interoperating *QTango* elements).
- The hardware referent, who commissions the requests to the GUI developer. He may draw a draft of the application containing the relevant controls and widgets and discuss every aspect with the application programmer.

WHAT MANGO IS NOT

Mango is not a tool to design complex control panels, with any other logic than the simple one offered by each *QTango* and *Qt4* widget through the exported *Qt properties* and the *signal/slot* connections. Nevertheless, GUI generation is easy and the work done with Mango is scalable and in no case is wasted: future improvements of the interface and the introduction of complex logic can be started from the *xml* generated with Mango by importing it into a *Qt4 designer* project.

GUI COMPONENTS

The Mango graphical user interface is made up of the following components (see Fig.1).

- An *Object Factory*: a list of *QTango* and *Qt* widgets ready to be used in the *main Mango widget area* by dragging and dropping them from the *Object Factory* widget.
- A *Tango DB Browser*: a tree representing the *Tango* devices exported to the *Tango* database. Attributes and commands of each exported device can be dragged from the *Tango DB Browser* and dropped into the *QTango* widgets populating the *main Mango widget area* component.
- The *main Mango widget area*: a container where *Qt* and *QTango* widgets can be dropped either from the *Object Factory* component or from other *QTango* panels on the desktop. Once the widgets are dropped into this area, they can be *selected, moved and resized* with the mouse, and their properties can be personalized through the *Object Properties* component on the left and by right clicking on them. Multiple selection is possible by pressing the *Control* key while clicking with the mouse.

* This work was supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3.

Currently, setting widget properties on multiple selections is not supported, but a bunch of actions is available by right clicking over a widget belonging to the multiple selection: move, resize, alignment to grid... Main Mango widget resize events are propagated to children, that resize proportionally, unless the option *Ignore main window resizes* is checked on the *Design Options* component. The *grid* drawn over each Mango container serves as a maths squared exercise book to ease placing the collection of widgets into the main Mango widget area.

- The *Tango Point Information* component: *QTango* widgets use this object to display useful information about their state.
- The *Object Properties* component: allows to click on a property and change its value by either directly editing the property on the tree widget item or by clicking on the custom button that might appear according to the property type. Custom editing dialogues are provided for complex properties like fonts and colours.

DESIGNING A CONTROL PANEL

Composing a graphical control panel with Mango is as easy as dragging an object from the *Object Factory* component and dropping it into the *main Mango widget area.* Once a widget becomes part of the *main Mango widget area*, it can be configured through its properties. For instance, a Tango *source or target* can be associated to a *QTango* widget. To change object properties, one has to first *select* the object with the left button of the mouse. The object is highlighted with a light blue colour and its cyan border is sensible to mouse resize actions. Multiple selection is possible through the *control* key pressed in conjunction with the mouse button. A set of actions is available for multiple selections by right clicking with the mouse over one of the widgets.

Simple Widgets and Containers

Mango widgets can be divided into *simple widgets* and *containers*. Containers are special widgets that can contain both simple widgets and other containers. They propagate the resize events to their children and manage saving their properties. One can populate the interface both with simple widgets and containers without worrying too much about what kind of object she is using. As one can see in Fig.1, the *readers* are connected to *Tango* devices.

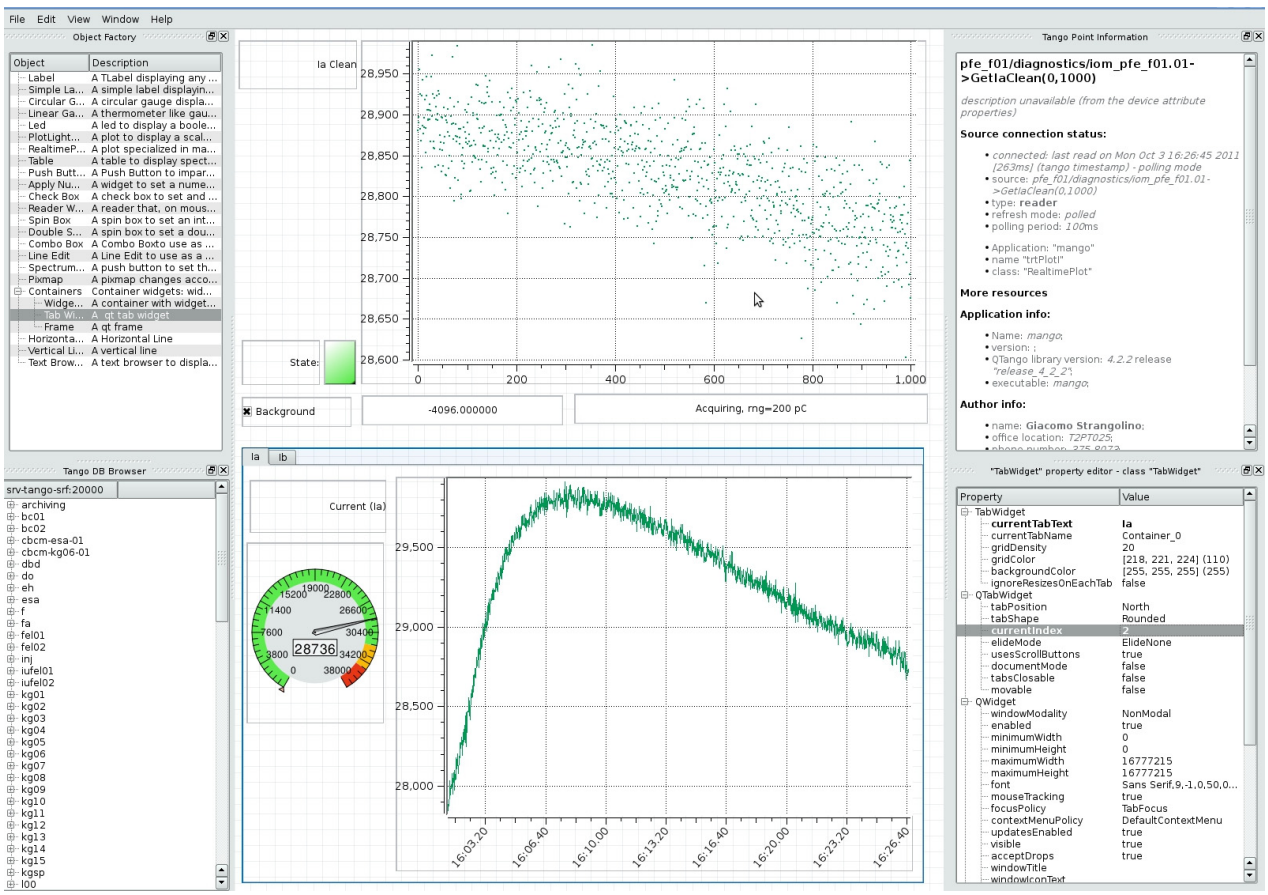


Figure 1: Mango graphical user interface.

In Fig.1 the *tab widget* is currently *selected* and its properties are accessible through the *Object Properties* widget. The *Tango Point Information* component displays the status of the underlying Tango connection that provides data to the plot. *Writers* are partially enabled: they cannot *write* the tango quantity they are connected to when in *design* mode.

EXECUTING A CONTROL PANEL

A Mango project saved as a profile can be loaded in two different ways:

- via the command line;
- launching the Mango application normally, then switching off *design mode* via the *View* menu and loading a profile by triggering the action *Profile Manager...* from the *File* menu.

In the first way you can integrate the Mango panel into a launcher icon or into an application browser. Fig.2 represents the same panel as in Fig.1 in execution mode.

PROFILES

The graphical interfaces realized with Mango can be easily managed via the Mango *profiles*. A profile stores the configuration of a GUI designed with Mango.

CONCLUSIONS

Mango is currently used by GUI and Tango device server programmers to sketch graphical user interfaces and testing purposes. Up to now, Elettra and Fermi@Elettra control room operators have designed and run with Mango about ten control panels.

REFERENCES

[1] G.Strangolino et al., "*QTango*: a library for easy tango based GUIs development", ICALEPCS2009, Kobe, October 2009, THP096.
[2] *Tango*, http://www.tango-controls.org/
[3] Qt for Application Development, http://qt.nokia.com/

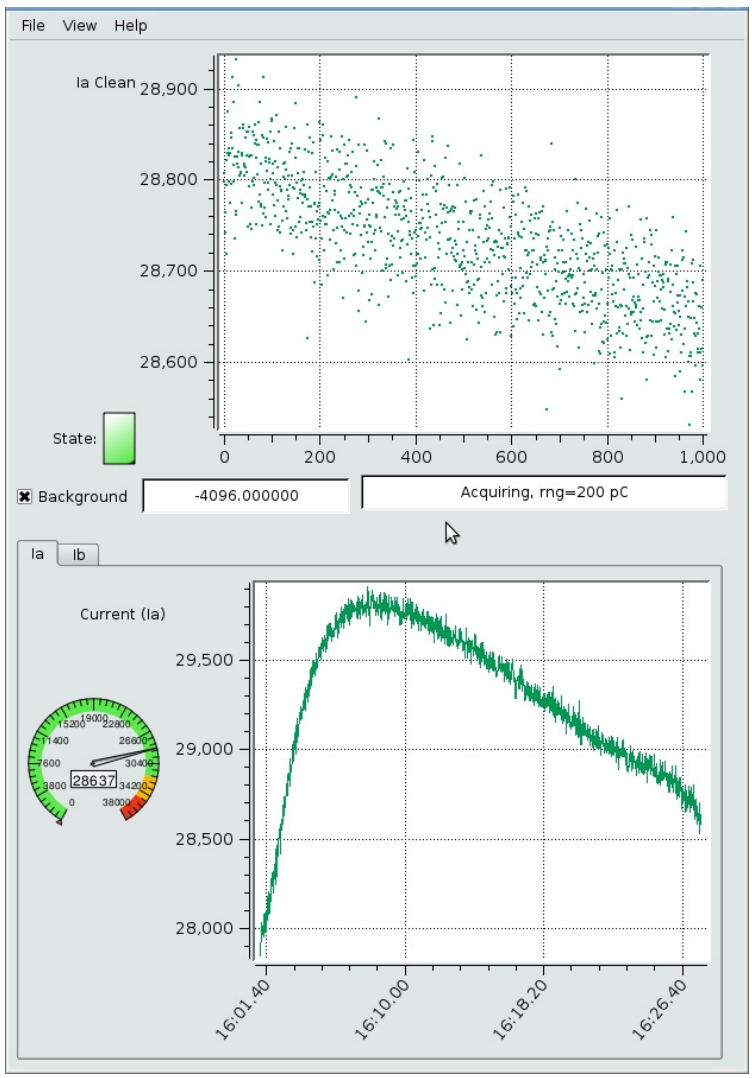


Figure 2: A Mango control panel in execution mode, currently used in Fermi@Elettra control room.

FURTHER DEVELOPMENTS IN GENERATING TYPE-SAFE MESSAGING

R. Neswold, C. King
FNAL†, Batavia, IL 60510, U.S.A.

Abstract

At ICALEPCS '09, we introduced a source code generator that allows processes to communicate safely using data types native to each host language[1]. In this paper, we discuss further development that has occurred since the conference in Kobe, Japan, including the addition of three more client languages, an optimization in network packet size and the addition of a new protocol data type.

INTRODUCTION

Writing software to support external messaging can be tedious and error-prone. The software author must make sure the required message fields are present, are aligned properly, and follow an agreed upon byte-ordering. If he cuts any corners while implementing the protocol, problems will appear – sometimes weeks or month later. Decoding packets takes extra care since the software must prove the sender's data is well-formed; in addition to byte-ordering and alignment issues, all fields must be validated and range-checked.

The protocol compiler is a command-line utility that generates the source code that safely serializes messages. The compiler takes, as input, a source file that describes the messages found in the protocol and creates the source files that implement it.

The benefits of using the compiler rather than hand-coding the protocol are:

1. Endian issues are handled automatically so the programmer only works with the native data types of their machine.
2. Packing issues are handled automatically so message fields are aligned properly.
3. The messages are implemented in a form that is native to the language (structs in C++, classes in Java, etc.)
4. Messages are completely validated (field values are range-checked, enumerations are validated, all required fields are present.)

Over the past two years, we've used the protocol compiler on many new projects. It has allowed us to focus all our attention on the application design because we know the communication layer just works. Despite it being a very useful tool, we have found a few minor shortcomings. The solutions to which are described in this paper.

```
enum Status { Away, Online };

request SendMessage {
    string text;
}

request UpdateStatus {
    Status newStatus;
}

reply NewMessage {
    string from;
    string text;
}

reply NewStatus {
    string who;
    Status status;
}
```

Figure 1: Protocol source used for examples.

ADDING ENUMERATION TYPES

While designing protocols, we noticed a particular idiom repeatedly being used. We would use an `int16` type to represent a field that could only have a limited set of values (the very use-case that justified adding enumerations to many programming languages.) Emulating these enumerations required us to write our own validation code to make sure the values sent and received were members of the limited set. But type-safety and field validation is exactly what the protocol compiler is supposed to do! So enumerated types were deemed necessary and added as an official data type.

In the generated code of each target language, the underlying integer for each enumeration value is based on a hash value, instead of being sequential, to prevent software from assuming any ordering of the values (and, hence, breaking later if the enumeration is changed.) Programmers are encouraged to ignore the underlying integer values and simply use the symbols provided.

NEW SUPPORTED LANGUAGES

Although the officially supported languages at Fermilab are C++ and Java, there are many groups that use alternate languages to write useful code that isn't meant for the Control Room operators. We've supported these efforts mostly by routing their data requests through Java servlets using the XMLRPC protocol. This approach works, but limits the clients' data rates and adds another "hop" to the return path. If we were to support these additional languages using the protocol compiler, they could request the data directly from the control system.

†Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

Table 1: Protocol Type to Language Type Mapping

Protocol Type	C++	Java	Python	Erlang	Objective-C
`bool`	`bool`	`bool`	`True` or `False`	atoms 'true' or 'false'	`NSNumber*`
`int16`	`int16_t`	`short`	`int`	integer with restricted range	`NSNumber*`
`int32`	`int32_t`	`int`	`int`	integer with restricted range	`NSNumber*`
`int64`	`int64_t`	`long`	`long`	integer with restricted range	`NSNumber*`
`double`	`double`	`double`	`float`	`float`	`NSNumber*`
`string`	`std::string`	`string`	`str`	`string` (which, in Erlang, is a list of integers)	`NSString*`
`binary`	`std::vector<uint8_t>`	`byte[]`	`bytearray`	`binary`	`NSData*`
`struct T {}`	`struct T {}`	`class T {}`	`class T`	tuple defined with a record specification	Objective-C class
`enum T {}`	`enum T {}`	`enum T {}`	`int` with restricted values	atoms	`enum T {}`
`optional T`	`std::auto_ptr<T>`	nil object references	presence or absence of class attributes	atom 'nil' or actual value	`nil` or actual value
`T[]`	`std::vector<T>`	`T[]`	list of T	list of T	`NSArray*` of objects

Table 1 shows the five currently supported languages and how the protocol data types map to their respective native data types.

The examples given in the following sections assume the protocol input file, `example.proto` (shown in Figure 1), was used.

Python Programming Language

There is a community of Python programmers at Fermilab which lean heavily on the XMLRPC protocol, so it made sense to add Python as target language.

Python's data types are less numerous than there are in C++, so the Python integer type serves several purposes. The generated code makes sure the values are in range before encoding or decoding messages. In addition, Python is a dynamically typed language. So, even though the generated code creates "official" protocol classes to be used for the messages, any class with the correct set of attributes can be used (attributes not defined in the protocol will not be part of the serialization.)

We chose to use the Python iterator interface as the mechanism for marshalling and unmarshalling messages. The unmarshal routines accept and read from a character iterator to decode the message while the marshal routines return an iterator which generates the character stream.

Supporting the example protocol in Figure 1 using Python first requires creating the module that serializes the messages. The following command creates the `example.py` source file:

```
pc -l python example.proto
```

After importing this new module in a Python script, a `SendMessage` message can be marshalled thusly:

```
msg = SendMessage_request()
msg.text = "Hello"
iter = marshal_request(msg)
```

The resulting iterator can be passed to anything that uses an iterator to obtain content: file objects; sockets; and even the bytearray constructor, if simply marshalling to a memory buffer is desired.

Unmarshalling requires a character iterator holding the incoming stream of data:

```
msg = unmarshal_reply(iter)
if isinstance(msg, NewMessage_reply) then:
    print msg.from, ": ", msg.text
```

The resulting message will be stored in `msg`. The specific type of message can be determined using `isinstance()`. If a message cannot be encoded or decoded, a `ProtocolError` exception is raised.

Objective-C

One line of mobile devices uses Java as its preferred development language, which we already support. But there is another line of popular, mobile products we also intend to use, so we added Objective-C as a target. It should be noted that this isn't a generic Objective-C generator; it uses objects defined in the OSX and iOS foundation frameworks and is, therefore, Macintosh- and iOS-specific.

The compiler generates two source files: a `.h` file containing the object interfaces and a `.m` file containing

the implementations. All message fields are represented by pointers to Cocoa objects (`NSNumber*`, `NSString*`, `NSArray*`, etc.), which follows coding conventions used in Macintosh and iOS development. Required fields in a message will always point to a valid Cocoa object and optional fields will either have a valid pointer or be set to `nil`. This generator encodes to and decodes from an `NSData*` object.

Converting the Python example to objective-C results in the following sample for marshalling a request:

```
example_SendMessage_request* msg =
   [[example_SendMessage_request alloc]
init];

msg.text = @"Hello";

NSData* bin = [msg marshal];
```

Since objective-C doesn't have namespaces or a way to nest classes, the names of the protocol messages need to carry enough information to make them unique and end up being quite long. However, objective-C code tends to be verbose, so long names don't stand out as much as they would in other languages.

To unmarshal data, the binary representation is loaded into an `NSData*` object which is then passed via the protocol's unmarshal method:

```
NSData* raw = [[NSData alloc] init];

// ...load the NSData buffer here...

example* msg =
    [[example unmarshal:raw] retain];

if ([msg isKindOfclass:
          [example_NewMessage_reply class]])
{
    example_NewMessage_reply* tmp =
        (example_NewMessage_reply*) msg;

    NSLog(@"%@: %@", [tmp.from],
[tmp.text]);
}
```

If there were any problems decoding the message, an `NSException*` is thrown.

Erlang

Recently, our department has taken an interest in the soft real-time, functional language Erlang[2] and we're looking for ways to use it. Before this can happen, though, Erlang needs to be made a first-class citizen in the control system by enabling it to communicate via ACNET and enabling it send and receive protocol messages. As of 2010, both those requirements were met.

The code generated for Erlang serializes protocol messages to and from Erlang binaries[1]. A binary is read in from a file or a network socket and sent to the unmarshalling function to be converted to an Erlang data type.

Table 2: Sample Integer encodings

Value	Encoding
0	0x00
1	0x01
-1	0xff
127	0x7f
-128	0x80
128	0x00, 0x80
-129	0xff, 0x7f

The Erlang language doesn't have the concept of a pointer so, to represent a missing optional field, its value is set to the atom `nil`.

Reaching back to our example, the Erlang version that encodes a message looks like:

```
Msg =
   #example_sendmessage_request{text="Hello"
},
Bin = example:marshal_request(Msg)
```

The resulting binary can be written to a socket or file. Like objective-C, Erlang doesn't have namespaces so identifiers tend to be verbose to ensure uniqueness.

Retrieving a value from a binary is as easy. In fact, it's so simple, we'll make this code segment do a little more than the previous examples. We'll use Erlang's pattern matching to show how we can decode both request types specified in Figure 1:

```
Bin = … %% obtained from socket, file, etc.
case example:unmarshal_request(Bin) of
 #example_sendmessage_request{text=T} →
   process_message(T);
 #example_updatestatus_request{newstatus=S}
→
   process_status(S)
end
```

OPTIMIZATIONS

Using the compiler has revealed a few opportunities for improvement. This section discusses a few techniques we used to improve both the encoded size and the performance of the encoders.

Packet Size Improvements

The encoded format uses integers to represent integers values, length fields and hashed field names. These integers were typically encoded using four bytes. We found that, most of the time, the length fields of strings and binaries were much smaller than the full range we allowed, resulting in many 0 bytes getting emitted. Integers values, too, didn't always push their range, so many 0 bytes (and FF bytes for negative integers) were emitted.

[1]The actual implementation isn't as symmetric as it sounds because messages are encoded as an Erlang "iolist" which is a binary or a list of binaries.

To reduce the superfluous data, we redefined the way integers are encoded. The current approach treats integers as the minimum number of bytes needed to encode a signed value. Some examples of the new encoding are shown in Table 2.

Unfortunately, the new encoding is incompatible with the older method, so the version of the header was bumped to version 2 (which guarantee messages with different integer encodings will fail when unmarshalling.)

Performance Improvements

When we originally wrote the protocol compiler, we focused on correct behavior before tackling performance issues. Now that we're satisfied with its reliability, we took a step back to see if we could improve the generated code.

It became clear that routines that emit integer values were also being called for integers that were constant (the hash values for message fields, for instance.) We replaced those calls at run-time with code that simply emits the corresponding constant byte sequence.

CONCLUSION

The protocol compiler is continuing to prove itself as an easy and robust way to get applications written in different languages hosted on different computer architectures to communicate. We have two active Erlang projects that are using the protocol compiler to access ACNET data at high data rates. We also used the protocol compiler output to deliver ACNET data to an iPhone/iPad application.

Since it takes an average of two weeks to support a new language, we're willing to expand the protocol compiler to support new languages that our community uses.

REFERENCES

[1] “Generation of Simple, Type-Safe Messages for Inter-task Communications”, R. Neswold, C. King, Proceedings of ICALEPCS2009, 2009, Pages 137-139

[2] Erlang Programming Language, 2011, Ericsson AB. <http://erlang.org>.

CAFE, A MODERN C++ INTERFACE TO THE EPICS CHANNEL ACCESS LIBRARY

J. Chrin, M.C. Sloan, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

CAFE (Channel Access interFacE) is a C++ library that provides a modern, multifaceted interface to the EPICS-based control system. CAFE makes extensive use of templates and containers with multiple STL-compatible access methods to enhance efficiency, flexibility and performance. Stability and robustness are accomplished by ensuring that connectivity to EPICS channels remains in a well defined state in every eventuality, and results of all synchronous and asynchronous operations are captured and reported with integrity. CAFE presents the user with a number of options for writing and retrieving data to and from the control system. In addition to basic read and write operations, a further abstraction layer provides transparency to more intricate functionalities involving logical sets of data; such 'group' objects are easily instantiated through an XML-based configuration mechanism. CAFE's suitability for use in a broad spectrum of applications is demonstrated. These range from high performance Qt GUI (Graphical User Interface) control widgets, to event processing agents that propagate data through the Object Managements Group's Data Distribution Service (OMG-DDS), to script-like frameworks such as MATLAB. The methodology for the modular use of CAFE serves to improve maintainability by enforcing a logical boundary between the channel access components and the programming extensions of the application framework at hand.

CAFE À LA MAISON

The Experimental Physics and Industrial Controls System (EPICS) is an established framework for the development of distributed control systems in the field of particle accelerators and other large-scale experimental endeavours [1]. A dedicated communications protocol, Channel Access (CA), allows for the high throughput of small data packets between the low-level hardware and external programs. Its native client library [2], written in the C programming language, provides remote access to controls data encapsulated in Process Variables (PVs) residing in EPICS Input Output Controllers (IOCs). It is thus the entry point for a number of CA interfaces to various C-based high-level programming languages, including declarative and 4^{th} generation languages such as Python and MATLAB respectively. While each of the resulting CA classes aims to achieve a similar interface and abstract layer, their context is typically confined to the system in which they execute. Another approach would be to enforce a logical boundary between the channel access components and the specifics of a given domain's C/C++ extension framework by providing a single C++ CA library for use across a number of C/C++ based programming languages.

There are several advantages to this library-based approach:

- The inherent simplicity and convenience of maintaining the channel access interface code and the flexibility it offers for library designers.
- New CA functionalities from future EPICS releases [3] need only be integrated in a single repository.
- Bindings to scripting and domain-specific languages are simplified.
- In-house (à la maison) CA expertise ensures a quick response to user needs and problem solving.

CAFE (Channel Access interFacE) is a C++ library that provides a modern, multifaceted interface to EPICS. It is intended to be expressive enough to provide the necessary abstractions in a convenient way and to be flexible enough to act as a host CA interface for other C/C++ based languages.

CAFE provides functionality for synchronous and asynchronous interactions for both individual and groups of channels. Data propagation through channel access is performed exclusively with native data types, although data presented to/by the CAFE interface may be in any meaningful format. This is also true of enumerated types. CAFE makes extensive use of function templates that allow identical code segments to perform a given task for any type of data. These have been used to implement client interfaces, underlying channel access classes, and data type conversions.

Callback functions have been implemented for all operation involving channel access connection handlers, event handlers and access rights handlers. Their invocation triggers their data to be written into a multi-index container provided by the Boost libraries [4]. The container takes ownership of the data object elements, which store all the current details of the associated PV, and provides multiple, distinct interfaces that allow for fast retrieval and modification of the element's data. This is exemplified by an inquiry to the current connection state of a channel which is a factor of $\sim 10^2$ faster compared to the native CA client call. While every gain in efficiency is welcome, it is, however,

the channel access data transfer time (∼ ms) that remains the dominating factor. There is, therefore, some leeway for code optimization within the CAFE Application Programming Interface (API). The most critical issue is to avoid dynamic memory allocation at each channel access transaction as this noticeably impacts performance. The optimal approach is thus for the created PV handle (object reference) to take responsibility for allocating the necessary memory for storing the handle's data set within the element of the container. Memory space is re-allocated whenever the connection event handler is invoked or upon demand.

CAFE aims to hide the low-level technical details of channel access as much as possible. It has thus been preconfigured with a reasonable set of default parameters that allow basic operations to be immediately undertaken without details of CA or other third-party APIs leaking into the application domain. Nevertheless, as users become accustomed to the many CA possibilities, they will wish to optimize connectivity in a manner that best suits the needs of the application, e.g. by influencing certain CA properties such as timeouts, and whether to instigate operations as blocking or non-blocking. Each handle possesses its own set of CA properties that can be easily configured 'on the fly' through designated container access methods. Thus, although the internal mechanics of certain channel access operations may change, the interface gratifyingly remains the same.

CAFE is also equipped with an abstract layer that addresses related data sets as single logical software units. Such data may be retrieved/set with a single method invocation. An XML configuration mechanism is used for the initialization of such CAFE 'group' objects. CAFE thus has its own XML schema for defining collections of related nodes and for constructing groups from collections and individual channels. While such an XML file consisting of collections may be manually instrumented, the practice is to extract nodes of the same type from a master XML accelerator file that defines the topology of the accelerator in Universal Machine Format [5]. A dedicated parser traverses the accelerator XML to identify collection members and to transform their data into the CAFE XML schema. The pre-defined collections are loaded from the generated 'Collections XML file' on initializing CAFE and may be addressed by interfaces through their identifier.

In constructing the channel access interface, effort had been made to follow sound practices presented in [6]. Any remaining errors, however, remain the authors responsibility. The work presented here has been undertaken in a Linux environment and in the context of the Swiss Light Source (SLS) and the 250 MeV SwissFEL Injector Test Facility at the Paul Scherrer Institute (PSI).

SAVEURS DE CAFE

CAFE's suitability for use as a channel access host within different applications is demonstrated with examples ranging from high performance Qt control widgets [7], to Event Processing Agents (EPAs) that propagate data through Object Management Groups's (OMG's) Data Distribution Service (DDS) [8], to script-like frameworks such as MATLAB [9]. Figure 1 illustrates how the CAFE API conceptually extends the different application environments.

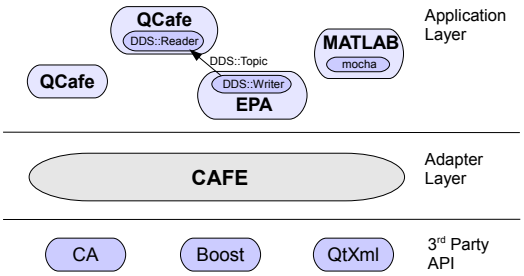


Figure 1: Application objects connecting to CAFE's adapter to channel access.

QCafe

The use of Qt for the development of CAFE-aware GUI (Graphical User Interface) widgets is a natural choice given the commonality in language. Qt offers components for a broad range of customizable widgets that can be readily applied to control system applications, and its graphics canvas and style engine allows modern, sophisticated user interfaces to be created. Qt's low latency coupled with C++'s runtime and memory efficiency further provides for a superior solution for building high performance GUI applications. These aspects could be well exploited for the computation and display of high-volume and high-rate data.

The QCafe package combines the speed, power and flexibility of C++/Qt with CAFE's extensive and robust channel access interface. The QCafe class diagram is shown in Fig. 2. The QCafe widgets are completely decoupled from CAFE as interaction with EPICS PVs is handled solely by QCafe's 'CAFE Interface Class'. EPICS data are then communicated between the interface class and the widgets using Qt's signals and slots mechanism. Some of the control/monitor widgets developed are shown in Fig. 3. The QCafeWheel widget is a customized widget built from other Qt GUI components and exhibits the same behaviour as its counterpart in MEDM (Motif Editor and Display Manager), one of EPICS established GUI builders [10]. Some of the widgets, namely QCafeStripChart, QCafeSlider and QCafeMeter, were developed with Qwt [11], a graphical extension to Qt.

Work is in progress to integrate QCafe widgets into Qt Designer, a tool for designing and building GUIs from Qt components. This will allow EPICS control GUIs to be created in an entirely code-free framework.

EPICS-Qt interfaces have also been developed independently at other facilities [12, 13].

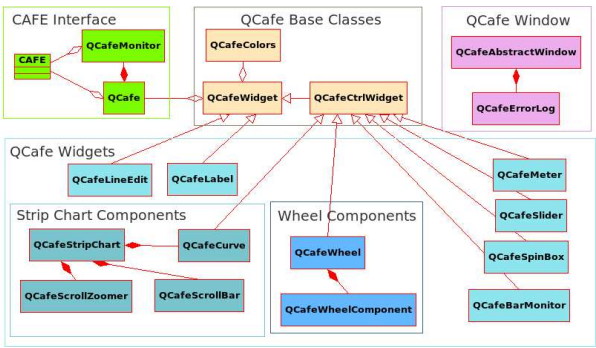


Figure 2: QCafe class diagram.

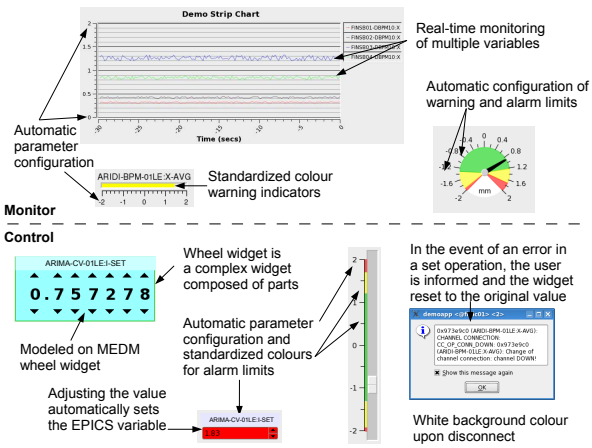


Figure 3: QCafe monitor and control widgets developed with Qt (v. 4.6.3) and Qwt (v. 5.2.1).

CAFE-DDS

High performance messaging middleware is an important component of distributed applications, and one that has been, for many years, an integral part of the high-level applications software infrastructure at the SLS [14, 15]. DDS is a recent OMG standard that supports high performance publish-subscribe messaging with configurable Quality-of-Service (QoS) guarantees. It has been put into good effect at the SwissFEL Injector Test Facility to disseminate summarized results of data acquired from the low-level hardware system. The use of CAFE combined with DDS (OpenSplice Community Edition, v. 5.1 [16]) is exemplified in an EPA that gathers data from the Digital Beam Position Monitors (DBPMs) [17].

The task of the DBPM-EPA is to aggregate, verify and analyze the DBPM data and to publish a summarized data set through a specially designated DDS Topic. The part of CAFE is to provide a simple, intuitive interface to initiate the acquisition of an entire complement of DBPM PVs. This is fulfilled by initiating monitors on the PVs through a single method invocation that includes, as input arguments, a group 'identifier' and a callback function that directs updated values into a designated data container. The DBPM PVs that comprise the group are defined in an XML configuration file by reference to a pre-loaded collection identifier (as described earlier) and the specified device attributes (Listing 1).

On parsing the DBPM-EPA XML configuration file, collections are expanded into their constituent elements and combined with the stated attributes to instantiate a CAFE 'group' object that holds the sequence of required DBPM PVs. The CAFE 'group' object created from Listing 1 is referenced in CAFE methods by its identifier, 'gDBPM'. Static DBPM data (names, positions, etc.) associated with each of the 'gDBPM' group members are extracted from the master SwissFEL XML file, and published to a separate DDS Topic for propagation to GUI clients.

The EPA is thus entirely configurable through XML, and the DDS Topics, which define and identify the data being propagated, provide sufficient information to allow for the dynamic configuration of a GUI client. Changes to the topology of the accelerator may thus be propagated to applications by simple modification to a configuration file, without need to recompile, as demonstrated in [5].

Listing 1: XML Configuration for DBPM-EPA at SwissFEL Injector Test Facility

```
<group id='gDBPM'>
   <collection> <id> cDBPM </id>
        <attribute> X </attribute>
        <attribute> Y </attribute>
        <attribute> Q </attribute>
        <attribute> POS–VALID </attribute>
        <attribute> Q–VALID </attribute>
   </collection>
</group>
```

CAFE Mocha

The use of MATLAB in the accelerator community has surged in recent years and is the principal choice of physics application developers at the SwissFEL Injector Test Facility [18]. (The Qt application presented in the previous subsection is one of a few exceptions.) Its increased usage, however, warrants a more extensive interface to the EPICS control system to the presently used mca (MATLAB Channel Access) package [19, 20].

CAFE mocha (MATLAB Objects for CHannel Access) is a MATLAB Executable file (MEX-file) that interfaces CAFE to MATLAB allowing CAFE routines to be called from within MATLAB in the same manner as MATLAB built-in functions. In many respects, the groundwork for MATLAB channel access connectivity had already been laid by mca. CAFE mocha, however, unlike mca, implicitly shelters underlying CA routines from the MEX API. Since the former are designated to the CAFE library, the resulting MEX-file is greatly simplified. Indeed, establishing communication between CAFE methods and the MATLAB workspace, where variables are passed into a function through input and output arguments, is largely re-

duced to mapping MATLAB data types to their equivalent CAFE/C++ data types.

The mocha interface improves on mca by providing support for all MATLAB data types, a richer set of access methods and a further physics oriented abstraction layer. Its use at the SwissFEL Injector Test Facility has been spurred on by the fact that certain tangible benefits became immediately available. Access to MATLAB primitive data types significantly improved performance in the analysis of camera data used in beam profile measurements. The advent of MATLAB's object-oriented programming capabilities has also allowed us to produce mocha-aware classes for specific operations, such as the acquisition of machine snapshots and orbit data (though some computations have been moved to within the MEX-file to enhance performance). Typically these objects, on initialization, extract static data, such as node positions, from CAFE's interface to an XML-based accelerator database (i.e. the master XML accelerator file). Object methods are also invoked to retrieve controls data with one synchronous group call, or to start monitors which cache updated values and, following the convention in mca, may optionally execute a MATLAB script to update a widget value. Widget updates may alternatively be handled through MATLAB's Event and Listener model, which allows a change in an object property to be monitored and a corresponding action to be triggered.

CAFE mocha's compilation on a 64-bit machine has also enabled a first comparison of the injector data with a computationally intensive three-dimensional space-charge simulation model.

A number of mocha mca-like MATLAB scripts [20], which is the usual way in which mca users access the mca MEX-file, have been provided to facilitate transition from mca to mocha for application developers.

HARD ROCK CAFE

Acceptance tests from several external APIs have been performed and give us confidence that CAFE is in a credible, 'Hard as Rock', state and can now be deployed. A short-term objective is to begin the task of replacing CA interfaces that are frozen to deprecated CA functions (as in [21]) with the CAFE API. Developments with QCafe continue with the aim to provide superior, high performance GUI applications, while CAFE-DDS offers a state-of-the-art publish-subscribe mechanism for applications geared towards a reactive form of programming. CAFE's mocha API has demonstrated immediate tangible benefits and work towards its consolidation draws near. The approach of having an extensive and flexible, C++ channel access library to serve as a host API for C/C++ based language extensions will lead to a thoroughly tested code base for other declarative and domain-specific environments, ultimately simplifying maintenance of code. If proved effective, then "Hard Rock Cafe's" mission, "to love all and serve all", is one that may be equally adopted by our very own brand of "Hard Rock CAFE" [22]!

REFERENCES

[1] EPICS, http://www.aps.anl.gov/epics/.

[2] J.O. Hill, R. Lange, "EPICS R3.14 Channel Access Reference Manual", http://www.aps.anl.gov/epics/docs/ca.php.

[3] A. Johnson, R. Lange, "Evolutionary Plans for EPICS Version 3", ICALEPCS 2009, Kobe, Japan, pp. 364–366.

[4] Boost Multi-index Containers Library, http://www.boost.org/libs/multi_index.

[5] J. Chrin et al., "XML Constructs for Developing Dynamic Applications or Towards a Universal Representation of Particle Accelerators in XML", IPAC 2011, San Sebastián, Spain, pp. 2295–2297.

[6] D. Zimoch, "Channel Access Client Programming", EPICS Collaboration Meeting, 27-29 July 2009, NFRI, Daejeon, Korea, http://www.aps.anl.gov/epics/meetings/2009-07/.

[7] Qt, http://qt.nokia.com.

[8] OMG-DDS, http://portals.omg.org/dds/.

[9] MATLAB®, http://www.mathworks.com.

[10] MEDM, http://www.aps.anl.gov/epics/extensions/medm/.

[11] Qwt, http://qwt.sourceforge.net.

[12] S. Baek et al., "KSTAR Widget Toolkit using Qt Library for the EPICS Based Control System", ICALEPCS 2009, Kobe, Japan, pp. 146–148.

[13] A. Rhyder, A. Owen, G. Jackson, "Qt EPICS Development Framework", PCaPAC 2010, Saskatoon, Saskatchewan, Canada, pp. 30–32.

[14] M. Böge, J. Chrin, "Developments to the SLS CORBA Framework for High Level Software Application", ICALEPCS 2005, Geneva, Switzerland, paper ID: WE4A.1-5O.

[15] M. Böge, J. Chrin, "An Event Service for the Propagation of Data", SLS Note: SLS-TME-TA-2004-0255, Dec. 2004, http://ados.web.psi.ch/slsnotes/tmeta040255.pdf.

[16] OpenSplice, http://www.opensplice.com.

[17] J. Chrin, G. Prekas, "A Taste of CAFE", ICALEPCS 2009, Kobe, Japan, pp. 821–823.

[18] M. Dach et al., "Control System in SwissFEL Test Injector Facility", ICALEPCS 2011, Grenoble, France, *These Proceedings*.

[19] T. Terebilo, "Channel Access Client Toolbox for MATLAB", ICALEPCS 2001, San Jose, California, USA, pp. 543–544.

[20] MATLAB Channel Access (mca), http://sourceforge.net/apps/trac/epics/wiki/MatlabChannelAccess.

[21] J. Chen et al., "CDEV: An Object-Oriented Class Library for Developing Device Control Applications", ICALEPCS 1995, Chicago, Illinois, USA, Paper ID: M4B-a.

[22] J. Chrin, "Hard Rock CAFE", EPICS Collaboration Meeting, 3-7 October 2011, PSI, Villigen, Switzerland, http://indico.psi.ch/event/EPICS.

EVALUATION OF SOFTWARE AND ELECTRONICS TECHNOLOGIES FOR THE CONTROL OF THE E-ELT INSTRUMENTS: A CASE STUDY

P. Di Marcantonio#, R. Cirami, I. Coretti, INAF-OATs, via G.B. Tiepolo, 11, I-34143, Trieste, Italy
G. Chiozzi, M. Kiekebusch, ESO, Garching bei Munchen, 85748, Germany

Abstract

In the scope of the evaluation of architecture and technologies for the control system of the E-ELT (European-Extremely Large Telescope) instruments, a collaboration has been set up between the Instrumentation and Control Group of the INAF-OATs and the ESO Directorate of Engineering. The first result of this collaboration is the design and implementation of a prototype of a small but representative control system for an E-ELT instrument that has been setup at the INAF-OATs premises. The electronics has been based on PLCs (Programmable Logical Controller) and Ethernet based fieldbuses from different vendors but using international standards like the IEC 61131-3 and PLCopen Motion Control. The baseline design for the control software follows the architecture of the VLT (Very Large Telescope) Instrumentation application framework but it has been implemented using the ACS (ALMA Common Software), an open source software framework developed for the ALMA project and based on CORBA middleware. The communication among the software components is based on two models: CORBA calls for command/reply using the client/server paradigm and CORBA notification channel for distributing the devices status using the publisher/subscriber paradigm. The communication with the PLCs is based on OPC UA, an international standard for the communication with industrial controllers. The results of this work will contribute to the definition of the architecture of the control system that will be provided to all consortia responsible for the actual implementation of the E-ELT instruments. This paper presents the prototype motivation, its architecture, design and implementation.

INTRODUCTION

The mock-up instrument, to be controlled, is a kind of multi-object (optical) spectrograph composed by the following main subsystems and components:

- One spectrographic arm with an ADC (Atmospheric Dispersion Corrector) and a dedicated CCD;
- One imaging arm with a filter wheel and a dedicated CCD;
- One calibration system equipped with a simple on/off lamp (e.g. Thorium-Argon lamp) and one characterized by a warm-up time required to reach the necessary stability (e.g. Deuterium lamp).

The skeleton of this prototype is based on a real astronomical spectrograph that is being built at INAF-OATs premises and is representative of the components that we expect to be part of a real E-ELT instrument.

#dimarcan@oats.inaf.it

FUNCTIONAL REQUIREMENTS

A detailed set of requirements is a starting point to produce both adequate software and electronic architecture design. For our specific prototype we reused the work done both by ESO and INAF-OATs as part of the studies for the E-ELT Phase A instruments. Dedicated use cases were developed covering aspects like system/instrument start-up, simulation levels, handling of logs and errors, and, typical of an astronomical instrument, handling of the acquisition, observation and calibration phases.

We have reused, moreover, some successful paradigms employed in the VLT software instrumentation framework ([1] and reference therein) applying the experience gained during the software development for the VLT instruments.

Command/ eply

Sub-systems and, in general, devices of the system respond to commands (e.g. *SETUP*) which execute specific actions. Each command is composed by a string made up of one or more keywords (in FITS-like format) and parameters chained together (e.g. *INS.FILT1.NAME U*). Each keyword specifies the subsystem and device the message is addressed to.

State Machine

Devices and components needed to operate the prototype follow the typical VLT state machine. The states implemented are: OFF (not running), LOADED (started-up), STANDBY (software initialized), ONLINE (hardware initialized) [1].

Parallelism

Operations in our prototype are executed in parallel. At the device level this means that all devices are capable of moving in parallel to spare time and minimize initialization procedures. At the higher level (i.e. interaction with the user) it means that the corresponding sub-system does not block upon receiving a request.

SOFTWARE ARCHITECTURE

The baseline design for the prototype Control Software architecture follows a VLT-like approach [1]. Several blocks/packages constitute the overall architecture and could be summarized as follows:

- Low-level control package (identified as **ICS** – the *Instrument Control Software*), responsible for handling the vital part of the prototype (motorized devices and lamps in our case);

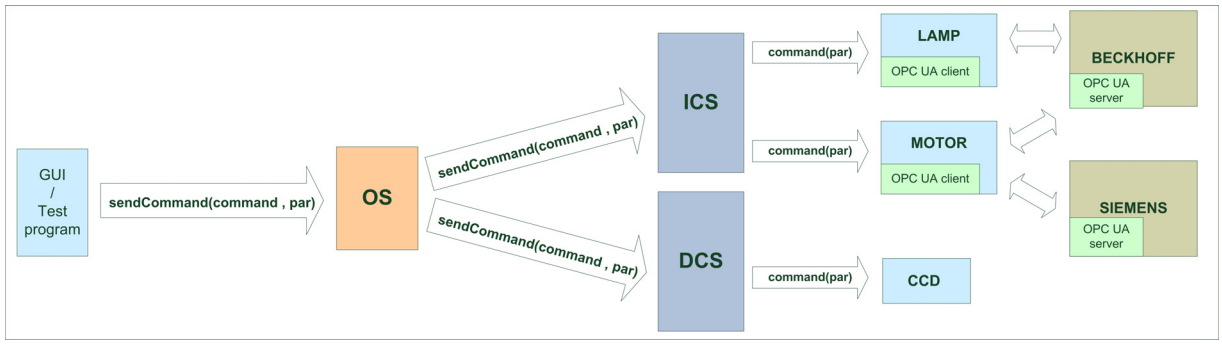


Figure 1: Mock-up prototype software architecture.

- High-level control package (identified as **OS** – the *Observation Software*), responsible for the coordination of activities of the detectors, low-level part and the telescope;
- Detector Control Software (**DCS**), responsible for controlling/handling all detectors.

We decided to use the ALMA ACS framework [2] as the infrastructure software to build our mock-up prototype instrument control software. The reason of this choice is essentially twofold:

- to use a modern, fully fledged, application framework to speed-up the prototype implementation
- to evaluate ALMA ACS complexity (learning curve), completeness, performance and ease-of-integration with other technologies in a view of its possible usage inside the E-ELT project.

The ACS philosophy is to model the system as a set of collaborating components located inside one or more containers contrary to the VLT case where the various software packages are implemented as a set of standalone processes.

Figure 1 shows the final prototype architecture. It is just a schematic picture and does not show all the involved entities (e.g. call-backs, threads). A GUI/test program sends a command to OS, which forwards it to the involved subsystems. Then the subsystem (at present ICS or DCS) sends the command to the appropriate devices. In the case of ICS, the connection between the ACS Components (LAMP or MOTOR) and the hardware (Beckhoff or Siemens PLCs) is realized through the OPC UA technology [3].

In the current version of the prototype the following C++ (ACS) Containers have been configured:

- *OSContainer*
- *ICSContainer*
- *DeviceContainer*, which hosts the LAMP1, LAMP2, FILT1 and ADC1 ACS Characteristic components;
- *DCSContainer*, which hosts the CCDIMAG and CCDSPECTRO ACS Characteristic components.

The motivation of having (so many) different containers even though the system is quite simple is to model it as a highly-distributed system exploiting the great flexibility of the ACS Component/Container model. This will facilitate future modifications. For example, OS and ICS, first implemented in C++, have been in parallel implemented in Java and deployed on a dedicated Java Container, in order to assess the feasibility of using this language for high level coordination tasks.

OS and ICS General Characteristics

In this prototype, the bootstrap of the system uses ACS activation on demand: each component is activated automatically only when requested by another component or by a client. In this way, if there is a need to interact with a specific device it can be just connected to a GUI and the corresponding component will be started up. On the other hand, if there is a request to activate the top-level OS, it will hierarchically activate ICS which in turn will be responsible to activate the needed devices. The ACS configuration database contains the needed deployment information.

The system is designed and implemented to allow executing operations fully in parallel. At the sub-system level (ICS, OS, DCS), command/parameters pairs are received (using a standard SETUP-command syntax implemented via CORBA method invocation) and the corresponding operations executed. Operations are non-blocking i.e. after checking the correctness of the received parameters the involved Component is free to accept new requests. At the OS level this is achieved via call-back mechanism. Before dispatching the incoming command to ICS or DCS, a call-back is instantiated and passed, together with the command, to the appropriate subsystem. In this way, it is responsibility of the involved subsystem to notify when the action has completed allowing OS to handle new requests. Once notified, OS propagates the answer back to the client (see Figure 2).

At the ICS level, a multi-threading paradigm is used instead. Parameters are parsed and sent to the appropriate device using a dedicated thread. Parallel actions are handled by separate threads and a thread join is used as a rendezvous point when all actions are completed (see Figure 2). This allows ICS to receive new commands if required (e.g. an emergency STOP or a SETUP on different devices).

The reason to use the call-back mechanism only at the OS level is due to the fact that OS (in principle) manages a limited number of sub-systems. Call-backs inside ACS are basically CORBA objects. They are therefore somehow "heavy" to activate and this could, at length, lead to some performance penalties.

Adopting these two different paradigms allowed us to compare the two options and to evaluate the easiness of their usage. The final impression is very positive: at the development level both paradigms are fully supported by ACS and also from performance point of view they behave as expected (see final section).

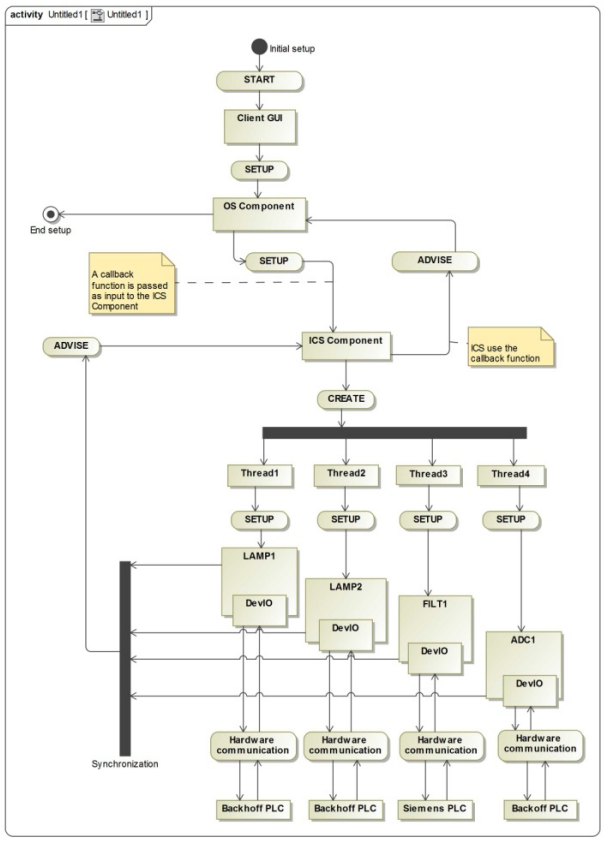


Figure 2: Activity diagram (only ICS part is shown).

The Java implementation for both OS and ICS components has been developed following for both cases the ICS multi-threaded architecture described above. This implementation has demonstrated that Java development is very convenient for these applications and has allowed us to demonstrate also the re-use of other system elements developed for ALMA (like the standard ALMA Master Component state machine or ALMA operator GUIs).

Devices and OPC UA Standard

Both ICS and DCS control hardware devices. Connection to the ICS PLC hardware is handled via OPC UA [3]. The OPC UA is a platform-independent standard through which various kinds of systems and devices can communicate by sending messages between clients and servers over TCP networks. In an OPC UA application we can usually find the following components:

- a "server", usually a thread on the process controller which exposes process data and methods that might run on another node and communicate with the hardware with proprietary protocols. It is supplied (typically) by the hardware manufacturer;
- a "client" that, by means of APIs, implements the OPC UA communication stack and allows the user application to access the data and methods exposed on the "server" side.

In the current prototype version the devices controlled by ICS are:

- LAMP1 – a simple on/off device (mimic a true ThAr lamp) controlled physically by a Beckhoff PLC;
- LAMP2 - a simple on/off device (mimic a true Deuterium lamp), with functionalities similar to Lamp1, but with a warm-up time i.e. the time that the lamp needs to reach full operative mode. Also this lamp is physically controlled by a Beckhoff PLC;
- FILT1 – a filter wheel with certain number of discrete positions controlled physically by a Siemens PLC;
- ADC1 – rotational device physically controlled by a Beckhoff PLC.

DCS controls the following devices:

- CCDImag – implements the interface to a Finger Lakes Instrumentation (FLI) CCD USB Camera.
- CCDSpectro – implemented with dummy responses.

We implemented our own OPC UA client inside the ACS by means of DevIO abstraction layer [2] using the OPC UA client provided by Unified Automation. The OPC UA SDK library is well wrapped inside just two classes: the first one implements all the steps that are necessary to correctly access (i.e. read, write, subscribe) a specific node through a DevIO interface; the second one handles all the connections/disconnections with the PLCs.

Another project based on ACS [4] is now taking this implementation as a starting point for writing a similar implementation on Java. Once this will be done, we will retrofit this prototype with a Java implementation of Device, to verify if also this lower level control part can be implemented conveniently in Java.

ELECTRONIC ARCHITECTURE

One of the aims of the prototype has been to evaluate possible hardware technologies that in terms of performance and cost could be employed in the actual implementation of the E-ELT instruments. Following the description given in the preceding sections (usage of PLCs and communication through OPC UA standard) two complete hardware infrastructures have been set-up to this purpose at our laboratories: one based on Beckhoff TwinCAT SoftPLC platform and the other on Siemens S7-300 PLC platform. Specifically the Beckhoff system is a CX9000 family embedded system equipped with I/O modules (EL1002, EL1014, EL2004) and the AX5201 servo drive able to control the AM3022 brushless synchronous servomotor. The Siemens system is a Compact CPU (CPU 314C-2 DP) equipped with the FM354 Servo Function module able to control a custom made DC drive used in ESO/VLT instruments.

In the Beckhoff case, the PLC software has been derived from the code developed for the ESO/VLT PIONIER instrument and uses the Motion Control (MC)

library conforming to PLCopen IEC 61134-3 as interface to the motors. In the Siemens case, due to the lack of PLCopen compliant libraries (cost and compatibility with our current hardware layout is under evaluation), dedicated code has been developed based on the Siemens FM354_354 block library. Note that the same subset of process variables as the one used in the Beckhoff implementation were exposed making the two platforms fully and transparently interchangeable. This is for sure one of the major achievements of this project: the prototype has shown that thanks to the ACS DevIO abstraction layer and a clever usage of the OPC UA standard, the overall system is unaware of the employed underlying technology/vendor products. By only changing few (ASCII) configuration files, but without having to recompile high-level ACS Components code or changing GUIs, it has been possible to control motors/lamps using platforms of the two different vendors with the possibility to interchange them in a completely seamless way. For the E-ELT instruments this has a quite big impact; it means that it could be possible to change the underlying low-level technology (e.g. due to obsolescence) without affecting the high-level framework or vice-versa in fully transparent way.

Performance

Performance measurement has been done with a mock-up for an ADC connected to the the Beckhoff platform. An ADC is a device able to compensate (counter-balance) the dispersion produced by the terrestrial atmosphere on the incoming starlight, composed by prisms that must be free to rotate. Their position depends on the celestial position of the object to be observed and must be therefore continuously updated.

The performance test that has been carried on is relatively simple: within a thread (spawned by the ACS Component Device associated to the ADC), the new position is calculated (simulating a real astronomical object) and then two calls are made trough OPC UA. The first call updates the target position; the second triggers the actual movement of the motor. To try to reach the real limit of the system, no check is performed on the positioning by the high-level software. In the case the motor is still positioning while a new "move" request is triggered, the target position gets simply updated and the motor continues its movement.

Measurements were made by time stamping, within the ACS code, when a thread is entered, before the first OPC UA call and at the thread exit.

At the PLC level, a physical output was mapped to the "move" flag and then sampled with an oscilloscope. No other tools where accessing the PLC by network when the test was made.

Analysis of these measurements shows that, as expected, most of the time is spent in the OPC UA communication (astronomical computation and thread life cycle management by ACS is negligible compared to the OPC UA calls). With our slow CX9010 system (consider that this is the cheapest Beckhoff CPU family able to run an OPC UA server) and a 10 ms PLC cycle the average time between two consecutive thread call is of the order of 80 ms as confirmed also in Figure 3 by an oscilloscope screen capture.

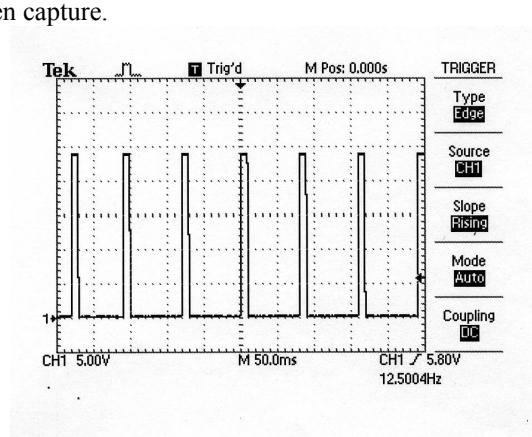


Figure 3: oscilloscope screen capture. Average time between two successive "move" commands is of the order of 80 ms.

Results can be certainly improved using a more performing CPU family (e.g. CX1030), but the actual measurements confirm already now that our (relatively) cheap hardware system connected to the ACS framework through OPC UA is able to sustain cycles of the order of 10 Hz which is well within the requirements asked for our applications.

CONCLUSIONS

The main outcome of our work is that the selected technologies fulfil the imposed requirements and are feasible alternatives to implement instrument control software for the E-ELT. ACS strong points are for sure the Component/Container paradigm, the transparent managing of the Component lifecycles, the DevIO interfaces and the various services offered by the framework which ease implementation (e.g. threads).

The major strength in using OPC UA is the transparent hardware management. Measurements show that performance achieved at the communication level are within specification for most astronomical requirements and therefore OPC UA, in perspective, could be further evaluated as an appealing technology to be employed for the future E-ELT instruments.

REFERENCES

[1] M.J.Kiekebusch et al., "Evolution of the VLT instrument control system toward industry standards", Proc. SPIE 7740, 77400T (2010)

[2] G. Chiozzi et al., "ALMA Common Software (ACS), status and development", ICALEPCS, TUP101, (2009)

[3] Mahnke et al., "OPC Unified Architecture", ISBN 978-3-540-68898-3, Springer (2009)

[4] I. Oya et al., "A Readout and Control System for CTA Prototype Telescope", ICALEPCS, MOPMU026, (2011)

A C/C++ BUILD SYSTEM BASED ON MAVEN FOR THE LHC CONTROLS SYSTEM

J. Nguyen Xuan, B. Copy, CERN, Geneva, Switzerland
M. Dönszelmann, Bogazici University, Istanbul, Turkey

Abstract

The CERN accelerator controls system, mainly written in Java and C/C++, consists nowadays of 50 projects and 150 active developers. The controls group has decided to unify the development process and standards (e.g. project layout) using Apache Maven and Sonatype Nexus. Maven is the de-facto build tool for Java, it deals with versioning and dependency management, whereas Nexus is a repository manager. C/C++ developers were struggling to keep their dependencies on other CERN projects, as no versioning was applied, the libraries have to be compiled and available for several platforms and architectures, and finally there was no dependency management mechanism. This results in very complex Makefiles which were difficult to maintain. Even if Maven is primarily designed for Java, a plugin (Maven NAR) adapts the build process for native programming languages for different operating systems and platforms. However C/C++ developers were not keen to abandon their current Makefiles. Hence our approach was to combine the best of the two worlds: NAR/Nexus and Makefiles. Maven NAR manages the dependencies, the versioning and creates a file with the linker and compiler options to include the dependencies. The Makefiles carry the build process to generate the binaries. Finally the resulting artifacts (binaries, header files, metadata) are versioned and stored in a central Nexus repository. Early experiments were conducted in the scope of the controls group's Testbed. Some existing projects have been successfully converted to this solution and some starting projects use this implementation.

MAVEN

Motivation for Maven

The BE/CO group is migrating from an in-house build tool to Maven [1], the industry standards for managing Java projects. This tool is widely used among the Java community by about 60% of developers. More than 100'000 open source projects are available on Maven Central repository, millions of connections happen per year. In addition, Maven can be used seamlessly with many tools for testing, source management, issue tracking, continuous integration, and so on. Since 2003, the BE/CO Java team have been putting a lot of efforts into software quality and reuse of resources in cooperation with other departments [2].

As the C/C++ developers want to improve quality and testing among their projects, the opportunity of unifying the build tools has been taken so they could profit from the same infrastructure, tools and experience from the Java development teams.

Maven Basics

Maven's key features are dependency management and versioning. It tends to enforce standards by providing a uniform build cycle with well defined succinct steps. In short, Maven takes the sources code, resolves and downloads the needed dependencies, builds the files and publishes the output to a repository. The output is called an artifact; it can be a jar, zip, war, etc...

Maven is plugin based and therefore can be easily extended with plugins. The latter can be provided by the Maven core developers, contributors, or one can write its own one. Plugins cover a very wide range of functionalities and have various purposes, such as generating docs, downloading specific artifacts, etc...

As shown in Fig. 1, a project is represented by a XML file called pom.xml (Project Object Model). Its main two parts are project information for the current project and dependencies information.

```
<project>
    <groupId>cern.testbed</groupId>
    <artifactId>testbed</artifactId>
    <version>1.0.0</version>
    <description>...</description>
    <url>http://...</url>

    <dependencies>
        <dependency>
            <groupId>cern.japc</groupId>
            <artifactId>japc-ext-cmwrda</artifactId>
            <version>[2.0.0,3.0.0)</version>
        </dependency>
        <dependency>
            <groupId>cern.japc</groupId>
            <artifactId>japc</artifactId>
            <version>2.11.0</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.8.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Figure 1: Simple pom.xml example.

THE NAR PLUGIN

Motivation, JNI and C Libraries

Maven handles the build steps for Java programs very well. Since Java allows the coupling to C and C++ programs it seems logical to include the step of compilation and linking of JNI (Java Native Interface) [3] modules as part of the Maven build steps. To extend Maven to handle native (C, C++, objective-C, etc...)

languages the NAR (Native ARchive) plugin [4] was written. How this plugin handles the compilation of JNI and other native modules for different platforms as well as their distribution and their usage of Maven's dependency mechanism is explained below. The NAR plugin consists of multiple sub plugins each of which takes part in the NAR lifecycle.

NAR Lifecycle

As mentioned earlier, to build any product Maven runs through a sequence of build steps. These steps are defined as a build lifecycle and directly associated to a packaging definition. By default Maven comes with packagings/lifecycles for jar, war, ear and some more. To enhance the standard build steps in the jar lifecycle with native compilation and linking, extra steps (in bold) where defined in the NAR lifecycle. The packaging for this lifecycle is NAR and a simplified version is given below:

- nar-download
- nar-unpack
- compile, nar-javah, nar-compile
- nar-testCompile
- test, nar-test
- nar-package, jar
- nar-integration-test
- install
- deploy

Native Sources and Headers

Maven assumes standardization for its plugins. The sources and header files for native parts of the code need to be stored in predefined places, which can be redefined if necessary. Header files and c files are not stored in the same location to make it easier to distribute the headers without the sources. These locations are used by the nar-compile plugin but also by the nar-javah plugin which runs the javah compiler. All generated output is stored, as usual in Maven, in subdirectories of the target directory.

AOL and Properties

To distinguish different platforms, operating systems and compiler/linkers the NAR plugin uses an AOL (Architecture Operating system Linker) qualifier. This qualifier looks like "i386-Linux-gcc" on a Linux i386 platform with gcc, but could be further extended in the form i386-Linux63-gcc4.1. The qualifier is used to handle different distributions as well as for selection of options for different compilers. Options and other flags are specified in an aol.properties file which sits next to the pom.xml file. A property file makes more sense than trying to put everything into profiles in the pom.xml as the number of platform/compiler combinations can be fairly large. Properties are stored in the aol.properties file for specific AOLs in dotted notation, for instance:

x86.Windows.msvc.cpp.compiler=msvc
i386.Linux.g++.c.options=-Wall -Wno-long-long

Compilation and Testing

For Java the maven-compiler-plugin handles the flags such as debug, optimization and others. We chose to use the cpptasks library [5] as this library unifies flags, options and linker strategies across different platforms and compilers/linkers. Based on the AOL a compiler, linker and their default flags are retrieved from the aol.properties file. These can be overridden by a project specific aol.properties file. The nar-compile plugin handles the native compilation/linking phase for which the cpptasks library was extended to use multiple cores in parallel to speed up compilation.

The nar-test plugin runs unit tests against the created JNI or standalone library. The test plugin makes sure that all libraries can be found.

The NAR Format and its Attached Artifacts

To package created libraries, executable and object files for re-use by others the NAR plugin uses its own format, the NAR file. A NAR file is no more than a standard jar file containing object files, executables, libraries and or header files. Files are stored in a directory structure that includes the AOL specifier but also reflects if libraries are static or dynamic. In its unpacked form the NAR compiler plugin is able to pick up header files and refer to libraries. This is important if some other package depends on a NAR library.

Three artifacts are produced under normal conditions: a standard NAR containing all Java class files if there are any as part of the project, a *-noarch* NAR file containing all non-architecture (non-AOL) specific files, such as header files and a *-<aol>* NAR file containing AOL specific files such as libraries. The first NAR file contains a property file that refers to the other two, which is used by the nar-download plugin, see below. The two latter NAR files are attached artifacts to the first. As can be seen this set of NAR files is a native equivalent for Java's single jar file.

The NAR files are split up in *-noarch* and multiple *-<aol>* files to make generating them easier and to download only the ones one need for a developer on a particular platform.

Distribution, Install and Deploy

The NAR plugin relies on the standard maven-install and maven-deploy plugins to install and deploy the primary NAR artifact and its attached artifacts. Any mechanism of caching such as the use of Sonatype Nexus [6] works transparently. Any Maven repository server will just store NAR files and their attached artifacts as another type of packaging.

Dependencies on other NAR Libraries

The reason for creating NAR files is so one can make other projects depend on them. These projects need to be also of the NAR packaging type and can then declare NAR dependencies. Any dependency declaration will initiate a download of the primary NAR artifact by Maven

itself. This primary NAR artifact is stored in the local repository.

Downloading and Unpacking Locally

Once a primary artifact of a dependency is downloaded, it is inspected for the above mentioned property file to see what attached artifacts need to be downloaded. In the normal case a *-noarch* and a *-<aol>* artifact will be downloaded by the nar-download plugin and stored in the local repository. As these NAR files as such are no use to any compiler, they will be unpacked by the nar-unpack plugin in a subdirectory of the current projects target directory (the latest version of Maven supports concurrent access to the local repository directory, so unpacking can also be done there in the future, thereby sharing artifacts and gaining space). The unpacked NAR files reveal the header files and libraries of the dependency and can thus be used the nar-compile plugin. Include paths and library paths will be set up automatically.

Cross Talk with other Systems

Other systems exists to build native (and even Java) code. The NAR plugin tries to be open and to integrate with those systems. One can for instance fairly easily call "configure", "autoconf" and "automake" of the GNU build system [7], or just call "make" to build libraries the usual way and use the NAR files purely for distribution and dependencies, as is explained below.

EXTENDING THE NAR PLUGIN

At CERN

At CERN, some projects were successfully converted to Maven NAR, but it resulted in big XML configurations. Indeed to simply add a compilation flag, about fifteen XML lines are needed whereas only one line is required with Makefiles, thus C/C++ developers were not keen to abandon the flexibility of their current Makefiles. In addition to that, CERN projects rely on cross-compilation, which is not covered by Maven NAR out of the box.

Design

Therefore it was essential to modify the NAR plugin according to our needs. A hybrid solution has been favoured: separate the build tasks between Maven NAR and Makefiles. Maven NAR takes care of the dependency management and versioning and Makefiles are in charge of the compilation process. NAR lifecycle has been modified so the goal nar-compile calls a Makefile instead of calling a compiler. Then binaries are expected to be generated and to be published to a binary repository. In our organization we use Sonatype Nexus for Java and decided to reuse it for C/C++ projects.

Finally, in order to support our cross-compilation infrastructure used by our Makefiles, it was necessary to modify cpp-tasks to add our compilers.

Implementation

As shown in Fig. 2, several steps are needed to build a C/C++ project with the customized Maven NAR. The following paragraphs describe each step along with its detailed implementation.

Makefile Generation Phase

The usual nar-download and nar-unpack phases are run, but in addition after those NAR will generate a Makefile with compiler and linker options. This Makefile contains only the dependency information for a specific platform, therefore the chosen naming convention is *Makefile.dep.<aol>*.

Compilation Phase

Maven NAR simply execute the command “make MAVEN_BUILD=true”. By convention, a Makefile needs to be present next to the pom.xml and its default target has to compile the source code. It also needs to include the Makefile.dep previously generated and use the defined macros from it.

The output binaries have to be placed in specific folders. The agreed standard is to put the library in *build/lib/<aol>* and includes in *build/include*. In BE/CO, we tend to enforce platforms independent headers, but some teams required platform specific headers. In this case, these headers go in *build/include/<aol>* and the Makefile generator will add an extra –I flag accordingly.

Packaging Phase

Files will place in the right folders so nar-package can do its job. Thanks to the directory conventions, NAR knows where to pick up what.

Deployment Phase

In Maven terminology, deployment means publishing on a server to make it available to other developers. This phase has not been altered from the official NAR plugin.

Usage Example

At CERN, C/C++ developers are used to define a macro called CPU to define the target platform. Instead of typing the whole target platform, shortcuts such as *L865* or *ppc4* are used. The same shortcuts were kept when invoking Maven commands, but these shortcuts are expanded to the AOL standards as i386-SLC5-gpp.

BENEFITS AND APPLICATION

Benefits

Since the dependencies information is separated and automatically generated, the Makefiles are simplified, the developers do not need to be concerned about dependency management and versioning anymore, and ultimately they can keep their habits with their Makefiles.

The previous implementation using pure Makefiles remains compatible with Maven NAR. Makefiles are called with the flag *MAVEN_BUILD=true* from Maven

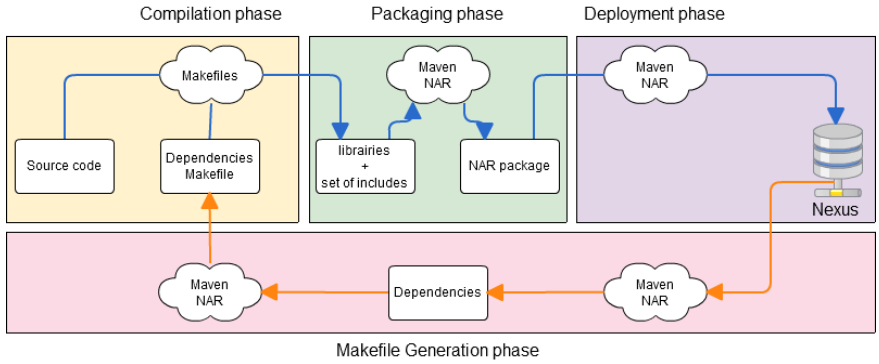


Figure 2: Extension phases.

NAR, thus it is known when a build is processed by Maven or pure Makefiles, some conditions can thus be added in order to include files accordingly.

Standards are also enforced, as explained earlier; Maven NAR needs to know where to pick up the different binaries (executables, libraries, headers). Directories need to have a well-defined structure with standardized names, as well as the files.

In addition of the build tool, the software development/release/deployment process and binary repository are also unified between programming languages. Java developers can easily switch to C/C++ and invoke the same commands through Maven to achieve the same goals. The build lifecycles are almost identical, first the code is compiled, then the unit tests are compiled and run, finally the product gets packaged.

Sharing C/C++ projects across and outside CERN becomes easy with Sonatype Nexus, CERN developers just needs to point to the binary repository and collaborators can proxy it and use the artifacts.

A CI (Continuous Integration) server takes care of building the projects with the bleeding edge source code from our SVN trunk. If a commit breaks a project, it will be immediately spotted.

Applications

The Maven NAR plugin is especially suitable to simplify cross-platform Java C/C++ build processes.

As an example, the build process of the CERN Data Interchange Protocol (DIP), a platform independent middleware protocol, was updated early 2010 to move from two entirely separate build systems relying on a series of manual steps and environment variables configuration into a unified Maven based build.

Because DIP is available both as a C++ and Java API, for three different platforms (Windows 32 bits, Linux SLC5 32 bits and SLC5 64 bits), a grand total of six builds, executed manually, had to be coordinated to assemble a complete distribution release.

Besides reducing the associated maintenance overhead, relying on a Maven build also helped to:

- Integrate more seamlessly the Java and C++ APIs (through JUnit testing)
- xDistribute its various components (Header files, static and dynamic libraries, auto generated documentation, associated development tools).

Since DIP is used by many projects at CERN, distributing it in NAR format also greatly simplified reuse for all CERN Maven based projects.

NEXT STEPS

Unit Tests

There are some improvements which can be done at the test phase. Instead of calling binaries which returns a code exit, we would like to integrate a testing framework to ease the writing of tests. Google C++ Testing Framework is a good candidate [8], to be used with Gcov [9] which offers code coverage. The goal is to generate full reports similar to the JUnit one, which will be displayed in our continuous integration server.

Merge Back with the Official Maven NAR

The CERN Maven NAR version embed functionalities specific to CERN, but most of the used methodology and chosen convention are standards in the C/C++ community. These changes need to be generalized and integrated back into the main version of Maven NAR in order to be able to profit from a community.

REFERENCES

[1] Apache Maven, http://maven.apache.org/
[2] B. Copy, M. Mettaelae, Agile Development and Dependency Management for Industrial Control Systems, WEPKS001, Proceedings of ICALEPCS'11, Grenoble, France.
[3] JNI, http://en.wikipedia.org/wiki/Java_Native_Interface
[4] Maven Nar plugin, http://duns.github.com/maven-nar-plugin/
[5] Cpptasks, http://ant-contrib.sourceforge.net/cpptasks/index.html
[6] Sonatype Nexus, http://nexus.sonatype.org/
[7] GNU Build System, http://en.wikipedia.org/wiki/GNU_build_system
[8] Google C++ Testing Framework, http://code.google.com/p/googletest
[9] Gcov, http://gcc.gnu.org/onlinedocs/gcc/Gcov.html

JAVA EXPERT GUI FRAMEWORK FOR CERN BEAM INSTRUMENTATION SYSTEMS

S. Bart Pedersen, S. Bozyigit, S. Jackson, CERN, Geneva, Switzerland

Abstract

The CERN Beam Instrumentation Group's software section has recently performed a study of the tools used to produce Java expert GUI applications. This paper will present the analysis that was made to understand the requirements for generic components and the resulting tools including a collection of Java components that have been made available for a wider audience. The paper will also discuss the prospect of using Maven as the deployment tool with its implications for developers and users.

INTRODUCTION

The CERN Beam Instrumentation Group (BI) belongs to the Beams Department (BE). Inside this group, the work of the BI Software Section is to implement real-time servers in C++ that control instruments developed for beam diagnostics located on all CERN accelerators and their transfer lines (LHC, LHC injectors, ISOLDE, LEIR, AD…). The team is composed of around ten physicists, engineers and students with good software skills. The main GUI clients are the hardware specialists in charge of the instruments, along with a few operators and accelerator physicists who use them during special manipulations for additional status and control. Before doing any low level C++ implementation, a design of the instrument is made using a dedicated CERN software architecture called FESA (Front-End Software Architecture) [1] (Figure 1).

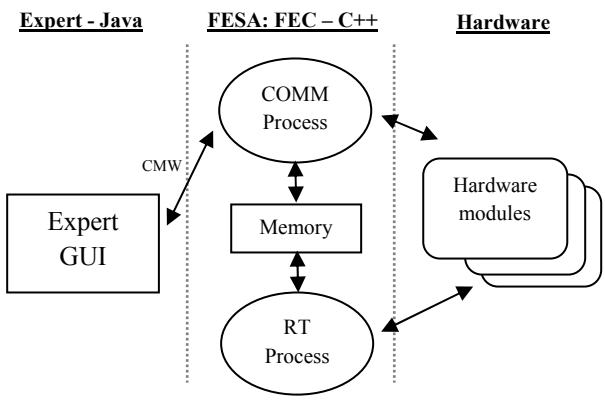


Figure 1: BE Front-End Software Architecture (FESA).

Providing graphical interfaces (GUI) implemented in Java is also part of the section's mandate. Hardware experts need to access their equipments in many different ways for parameter setting, signal visualization, error diagnostics, calibration, data post processing and so on.

Java (JDK1.6) is currently the BE department's development language for all GUIs and the software section's Java developments follow this standard. The low level software architecture, middleware and Java component libraries are provided by the BE Control Group.

PROBLEMATIC

Requests and specifications for new expert GUIs are often very different and depend on the type of instrument, the acquisition electronics and the type of diagnostics to be performed. Therefore, a standardization of a framework's functionalities and a static list of options is insufficient. The software engineer should instead be able to create ad-hoc Java classes and components (graphical or not) to fulfil specific requests.

Nevertheless, since the software requirements, although different, often have a similar overall structure, it made sense to have a common Java framework on top of which everyone could build specific applications. A functionality review of current expert GUIs revealed that a common customizable graphical framework is an efficient way to standardise and facilitate the development of a Java project implementation.

All of CERN's beam instrumentation systems belong to a particular accelerator domain (LHC, SPS, PS…) and are triggered by their own timing events. Each instrument is identified based on a database of devices that links it to a certain device name. The application window should therefore, for example, be able to display the incoming machine cycles and allow easy retrieval of device-names. Another important point is that the number of data types in an acquisition is limited. So, having the same graphical component to input or display a scalar or plot an array using common plot components would be a very efficient solution as it avoids code redundancy and strengthens software quality.

FROM THE OLD TO THE NEW FRAMEWORK

Why Did We Have To Change?

The main purpose of a common expert GUI framework is to provide a skeleton that can easily be built upon, so reducing the need to re-develop large sections of code for new applications and in so doing decreasing the total time spent to create such an expert application.

The old expert GUI framework, in place since 2004, was designed having the same goals in mind, but several issues have appeared over the years. People were therefore using this framework less and less and instead started developing their own applications from scratch. As already mentioned, an internal questionnaire revealed this was mostly due to the rigidity of the framework and the lack of desired functionality. Rigid, because this skeleton was providing a structure in the form of tabs and panels,

which was sufficient at the time when it was designed, but became a constraint as the panels to be shown always needed to be heavily customized. Several missing basic functionalities that were mandatory in terms of beam diagnostic tools also forced programmers to abandon the framework and instead implement their own code.

Besides these concerns, the section was frequently experiencing issues related the dependencies of their applications. All third party libraries coming from other CERN groups were directly obtained at runtime from external repositories and whenever a library lost its backwards compatibility, all the applications depending on it would break.

With the new expert GUI (v2), we tried to address these issues by providing a framework that still has all of the useful functionalities of the old framework, such as the communication and plotting packages, but is also programmatically more flexible (Figure 2). At the same time, we tried to manage the dependency issue by using a specific tool developed to handle this problem. Additionally, other useful components written by members of the section already existed and it was our intention to make these available to all users through the new environment.

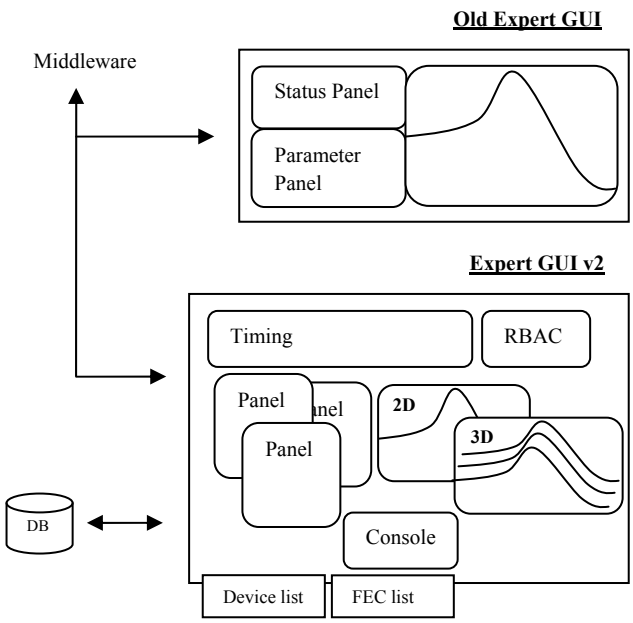


Figure 2: Old and new expert GUI.

What Did We Change?

The expert GUI framework needed a facelift both visually and programmatically. The main transformation was to convert the frame class that always had the same look with fixed tabs and panels to two different classes that inherit an abstract *ExpertGUIFrame*. First there is the *EmptyFrame* that basically provides an empty window where any of the desired components can be included. Second there is the *BasicFrame* which already contains commonly used components such as the RBAC (CERN Role Based Access Control) toolbar and a built-in console at the bottom of the frame.

In order to cover the need for some applications to look like the old expert GUI, a couple of panel classes were included in the new framework (*VintagePanel*, *VintageStylePanel*) that could be easily incorporated into the *BasicFrame*. If other predefined frame classes are desired in the future, they can be easily created with a framework extension.

In the old expert GUI, the very first step to create a program was to create a new main class that inherited properties from the framework main class. While this is still possible in expert GUI v2, by inheriting from the *EmptyFrame* or *BasicFrame* classes, the recommended way to obtain the framework functionality is to fetch a frame instance from a factory class and to use it as a delegate in your main class. This makes sure that the client who is using the provided frames only sees and uses the methods which were meant to be used by the class.

The communication library has basically stayed the same except that there is a new type of communication that deals with database (DB) connections. A new *CommunicationManager* class has been introduced so that the developer need not care about the details of instantiating different kinds of communications (FESA device or DB).

Applications often deal with more than one device, sometimes even with more than one instrument type. To facilitate the initialization of the communication and the handling of those devices, the factory class needs to provide a list of devices and databases. This information is used to create a menu item in the menu bar where all the relevant devices and databases are registered. Additionally, it feeds two manager classes that deal with the actions (button clicks) and the information that accompanies these devices.

Instead of managing the dependencies manually, as is done for applications based on the old expert GUI, we have tried to incorporate a third party tool called Maven, which automatically resolves all dependencies of a project and takes only the specified libraries from a local section repository.

This repository can also be used to share components between the members of the section simply by publishing their libraries into it.

What Are The Benefits?

The expert GUI v2 has become more flexible and it can more easily suit individual needs than its predecessor. For example, if someone wishes to completely write their own application from scratch, they can do so by instantiating the *EmptyFrame* but they wouldn't have to forego the communication library. Or, if someone wants to have an application that looks the same as the old expert GUI, it would also be possible to create it within the same framework. Finally, the new framework offers access to libraries (components) written by others, so everyone can profit from each other's work.

The usage of Maven should simplify the maintenance of the dependencies and generally facilitate the software build lifecycle.

JAVA SOFTWARE PROJECT MANAGEMENT WITH MAVEN

What is Maven?

In short, Maven is a software tool for project management and automation of builds [2], i.e. a build component testing and dependency management tool. Only a subset of the capabilities of Maven has so far been tested within the BE-BI Group.

Maven is not a graphical tool in its own right but a framework that is installed on the system (in an IDE i.e. Eclipse) which assists the development of a project. It is built around the concept of a "build lifecycle". This means that every complete project has to go through all the stages of the cycle which, for example, can contain compilation, testing, packaging (producing a jar), deploying in a local repository, deploying in a remote repository, etc.

The project has to successfully pass all of these stages. In a "traditional" project, these stages would need to be done separately. This manual mode forces the programmer to repeat the same procedure for each stage. Maven attempts to automate these procedures by extracting the pieces of configuration which are needed for every stage (for example: jar name, files to include in the jar, content of the manifest file, deployment location, etc..) and bundles them into a configuration file (Project Object Model) that is specific per project. The most important part of that file in our case is the list of direct dependencies.

Why Do We Need It?

We are not obliged to use Maven, and for us it started more as an experiment to see whether it could fit our needs. We were hoping to profit from the standardisation that comes with Maven as its capability for solving the dependency problem. Its concept of build lifecycle gives many structures almost for free, such as JUnit testing, packaging, JAR signing and generally structured project development. One additional strong argument to study Maven came from the fact that CERN Controls Group was also looking towards using Maven in the future.

Can We Use It?

Our deployment strategy does not implement any versioning of a given application with only a development and operational version at any given time. These two versions used to be deployed by an ANT script into two distinct folders. Each of these folders contains all the Java libraries (JAR) of all the applications, both development and operational versions. Either of these versions can be launched through a program called the *ApplicationLauncher* [3] (Figure 3).

A Maven project can be identified in a repository through three parameters: *GroupId*, *ArtifactId* and *Version*. These three qualifiers determine the folder structure in the local repository where the project is stored. However, if one decides to deploy using Maven, by default it forces you to use the same folder structure. It turns out that our current deployment strategy therefore does not fit well with this Maven approach, requiring some tricks to get rid of the Maven folder structure or significant modification to the *ApplicationLauncher*. In order to stay compatible with the current *ApplicationLauncher* it was therefore decided to opt for an Apache ANT script that copies the deployable JAR files to the desired location.

How Are We Using It?

Maven's dependency management system is declarative, transitive and simple to use. The libraries that a project needs to work properly are declared in its POM (Project XXX YYY) file. The three pre-cited qualifiers are enough for Maven to locate the right library in its repository system. Maven can provide an arbitrary number of repositories where it can obtain the requested libraries. It offers the possibility to specify a local repository which can be used internally in the section and not made public to other developers, allowing the possibility to control and manage changes i.e. updates of external JAR files.

The dependency system's transitivity (inheritance of properties and dependencies) facilitates the development of a project. Only direct dependencies need to be provided to Maven. If the libraries that a project relies on depend on another project, Maven will take care of this. It knows where to fetch the indirect dependencies, since all projects and libraries in the repository come along with a POM file that declares their dependencies.

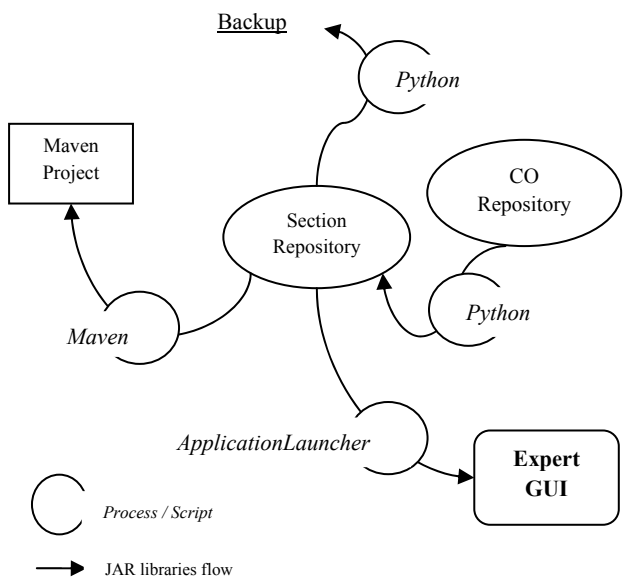


Figure 3: JAR libraries flow.

The inheritance concept, coming from object-oriented programming, is applied to the POM file. Through

inheritance, common settings and properties can be propagated from a "parent" project to projects inheriting from it. This mechanism gives the possibility to define a parent project that contains most of the dependencies and settings that are usually needed in any project. A creator of a new project therefore only needs a POM file to inherit from the parent POM file in order to have all necessary dependencies, to know the location of the keystore for signing JAR files and to have the settings in a manifest file, etc…

In the old expert GUI framework, one needed to check out a template project from the Apache Subversion SVN repository in order to start a new project. Maven also helps in this perspective because it provides a template through a mechanism known as "archetypes". An archetype gives the possibility to steer several properties of a project. It can impose a predefined folder structure on a new project providing opportunity to standardize structure across Java projects. It can also provide additional files such as ANT files or log files and can give a source code template for the entry point of a project.

What Are The Benefits?

It took a rather long time to set-up a generic environment to work with Maven and then to effectively use the tool for development. As it was not possible to deploy our JAR files the way we wanted, we were forced to opt for a hybrid solution using both Maven and ANT. The most comfortable way for us to develop an expert application is to have a simple ANT file that does the cleaning, the compilation, the packing and the final deployment with a simple mouse click as was the case for the old expert GUI. In order to achieve this and spare individual users from unnecessary Maven details, the Maven commands are encapsulated in such an ANT file, with all commands forwarded to the Maven system except for the deployment task which is done directly using ANT commands. To backup files and to retrieve specific JAR files, a Python script is used.

The main advantage of using Maven is the control of the dependency files. In addition the standardization of projects and the use of archetype templates greatly lower the effort required to start a new expert application project.

The incorporation and usage of Maven in our projects is an on-going process and there are still details to be tested. Most of the benefits could probably have been achieved in a different way using different tools. Many of these options, however, would have probably required an unacceptable amount of manual work and would have ended up in a patch work of different components needing to be unified to provide a similar functionality as Maven.

CONCLUSION

The analysis and improvement of our Java software development tools are still on-going. This process is taking time because of several constraints. Keeping backward compatibility with our old applications, sticking to CERN software standards, and covering the needs of all our programmers are just some examples of things to be considered.

The new expert GUI has already given very good results. Users can easily and quickly create a Java project with a pre-defined structure that will allow them to run an application in two mouse clicks. At the same time, they are able to add whatever components they need to libraries that are now common to all. These components have been specified inside our section and therefore already provide most of the functionalities that might be needed in such applications.

The use of Maven is not completed and has led to some integration problems for our Java software architecture. Nevertheless, the handling of the library dependencies and the archetypes are very useful. The CERN Controls group has not yet confirmed that Maven will be one of its standards, but from our experience so far, we will give positive feedback regarding these two interesting features.

REFERENCES

[1] Michel Arruat et al, "Front End Software Architecture [FESA]", http://accelconf.web.cern.ch/accelconf/ica07/PAPERS/WOPA04.PDF, ICALEPCS 2007.

[2] Maven: http://maven.apache.org/pom.html

[3] P. Karlsson, S. Jackson, "The introduction of hierarchical structure and application security to Java web start deployment", http://accelconf.web.cern.ch/AccelConf/ica05/proceedings/pdf/O4_008.pdf, ICALEPCS 2005.

EXPLORING A NEW PARADIGM FOR ACCELERATORS AND LARGE EXPERIMENTAL APPARATUS CONTROL SYSTEMS

L. Catani, R. Ammendola, F. Zani, INFN-Roma Tor Vergata, Roma, Italy
C. Bisegni, P. Ciuffetti, G. Di Pirro, G. Mazzitelli, A. Stecchi, INFN-LNF, Frascati, Italy
S. Calabrò, L. Foggetta, LAL-CNRS, Orsay, France & INFN/LNF, Frascati (RM), Italy

Abstract

The integration of web technologies and web services has been, in the recent years, one of the major trends in upgrading and developing control systems for accelerators and large experimental apparatuses. Usually, web technologies have been introduced to complement the control systems with smart add-ons and user friendly services or, for instance, to safely allow access to the control system to users from remote sites.

Despite this still narrow spectrum of employment, some software technologies developed for high performance web services, although originally intended and optimized for these particular applications, deserve some features that would allow their deeper integration in a control system and, eventually, using them to develop some of the control system's core components. In this paper we present the conclusion of the preliminary investigations of a new design for an accelerator control system and associated machine data acquisition system (DAQ), based on a synergic combination of network distributed object caching (DOC) and a non-relational key/value database (KVDB). We investigated these technologies with particular interest on performances, namely speed of data storage and retrieve for the distributed caching, data throughput and queries execution time for the database and, especially, how much this performances can benefit from their inherent scalability.

INTRODUCTION

Two main motivations support the decision to start investigating a new approach in the design and development of CS for particle accelerators.

New developments in this field, similarly to what has happened in recent years, will be basically directed towards improving their functionalities by introducing new services, or improving existing ones, to complement the basic features that are essential for the remote control of the accelerator's components.

These new capabilities rather than being accessorial will be, in many cases, fundamental for the optimal operation of new accelerators that will require careful tuning to achieve the desired performance. An example may be the data acquisition system that is intended to provide machine physicists, but also the experimental groups, with all the information needed to recreate the operational state of the accelerator (set-point of components, information from the beam diagnostic etc.) at any instant during the operations of the machine.

The analysis of recent developments on high-performance software technologies suggests that the design of new accelerator CS may profit from solutions borrowed from cutting-edge Internet services. To fully profit from this new technologies the CS model has to be reconsidered, thus leading to the definition of a new paradigm.

The second strong motivation for this development follow the recent approval, by the Italian Ministry for Education, University and Research (MIUR) of the construction of a new international research centre for fundamental and applied physics to be built in the campus of the University of Rome "Tor Vergata".

It will consist of an innovative very high-luminosity particles collider named SuperB [1] and experimental apparatuses, built by an international collaboration of major scientific institutions under the supervision of Istituto Nazionale di Fisica Nucleare. Clearly, it will offer great opportunities not only for new discovering in particle and applied physics, but also for breakthrough innovation in particle accelerators technologies.

THE !CHAOS FRAMEWORK

A typical example of software technology emerging from developments of Internet services is the class of non-relational databases known as key/value database. They offer an alternative to relational databases (RDMS) that is having a growing success and interest among developers of web services because of their high throughput, scalability and flexibility.

Another example is the object caching, distributed systems that are used to store, in the servers' RAM, frequently requested sets of information in order to both respond faster to requests and distribute the load of the main server to a scalable cluster of cache servers.

These two software technologies, clearly cited on purpose, represent the core components in the design of this new control system we named !CHAOS[2].

In particular, the KVDB is used by DAQ for storing what we call *history data*, while the DOC implements the service for distributing *live data* from the front-end controllers to clients replacing the client/server communication.

Compared to the typical structure of CS, usually represented by the so-called *standard model* [3] of control systems, in the !CHAOS data flow the client (top) and front-end (bottom) layers are not directly connected and, especially, data is not sent by controllers when triggered by

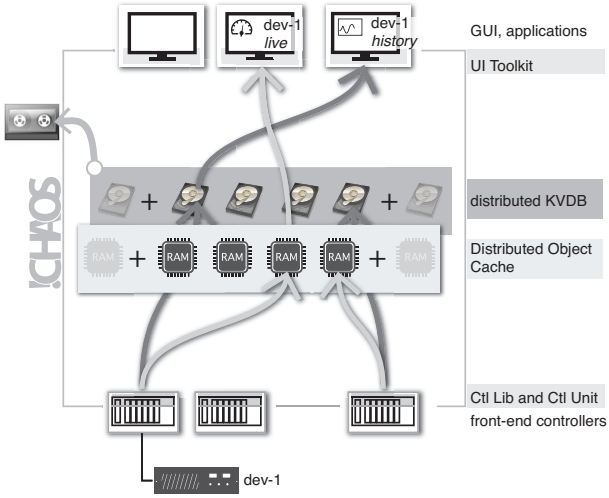


Figure 1: Data flow in the !CHAOS framework.

client request. Instead, alternatively to the typical point to point communication of network distributed systems, in !CHAOS *live data* flow from front-end controllers to the DOC servers, according to the independently adjustable refresh rate, from which datasets can be asynchronously read from any client.

This solution offers a number of advantages.

Firstly, we can use the same strategy, and topology, for both distributing *live data* and storing *history data* as shown in Fig.1. Datasets that need to be updated are identically pushed, by front-end controllers, to both DOC and KVDB servers by issuing *set* commands. It means that data collection mechanism for DAQ is inherently included in the !CHAOS communication framework because both *live* and *history* data are pushed by the data source (the front-end controllers) to similarly distributed caching and storage systems. Moreover, since both DOC and KVDB use key/value data storage, formatting and serialization of datasets can be identical.

Secondly, both the client applications and the front-end controllers are simple clients of the distributed object caching and DAQ. In particular, provided the DOC has an object container for each dataset of the CS, defined by its unique key, a GUI client simply send to the !CHAOS DOC service a *get* request for the object identified by that particular key, i.e. the dataset describing the associated device. On the other side the controller responsible for that device update its dataset, according to the push rate defined for it, by issuing *set* commands to the DOC.

Data refresh rates, as well as other meta-data and global parameters, will be managed by the meta-data server (MDS) that will be described later.

It's worth to underline that in !CHAOS the front-end controllers don't need to run servers for providing data to clients since they themselves are clients of the data distribution and storage services. That improves their robustness, portability and prevents front-end controllers from overload originated by multiple clients' requests.

A fundamental property of both DOCs and KVDBs is their intrinsic scalability that allows distributing a single service over several computers. Moreover, dynamical keys re-distribution allows automatic failover by redirecting to other servers the load of failed one. By taking advantage of this feature !CHAOS can be easily scaled according to both different size of the accelerator infrastructure and the performance required thus avoiding any potential bottleneck that may be expected as the weak link of the star-like communication topology.

In conclusion !CHAOS is a scalable control system infrastructure providing all the services needed for communication, data archiving, timing, etc. to which both front-end controllers and GUI applications plug-in to access and, to some extend, expand its functionalities.

Control Units: The !CHAOS Front-end

Fig.2 shows the logical structure of the software running in a front-end controller. The Control Library (CL) and the Control Unit (CU) are components of the !CHAOS infrastructure while the device management plug-ins (DMPs) are software drivers complementing the !CHAOS framework functionalities by providing the interface to the device. The development of these components is expected either as contribution, or under responsibility, of device experts.

One or more instances of CU can run simultaneously, though completely independent, in a front-end controller. Each of them will be dedicated to a particular device or a family thereof, specialized for that particular components by means of the device management plug-in. The latter are a set of routines implementing five main functionalities: initialization, de-initialization, control loop, dataset update and commands execution.

The Control Library, by means of its *managers*, will provide both the environment for the execution of the Control

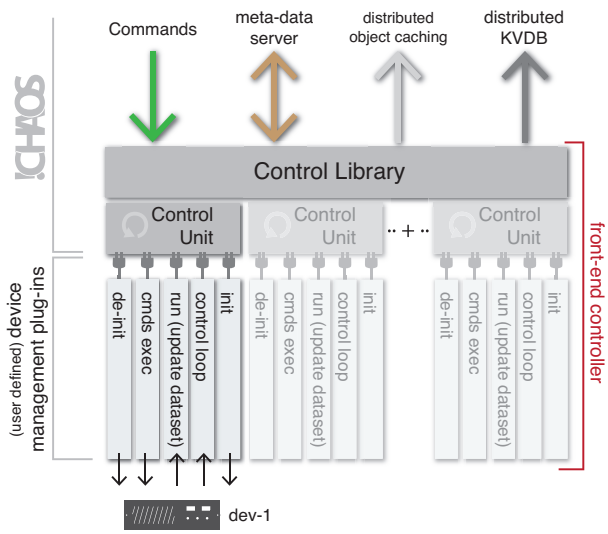


Figure 2: !CHAOS components for the front-end controllers.

Units and the functions needed to access the centralized services (DOC, KVDB, MDS).

Since the CU will control the execution of DMPs, the former will be responsible of invoking the *run* module, according to the refresh rate defined for that device, for reading the device status. The data returned will be used by the CU, via the Control Library, to feed the KVDB and to refresh the value of the correspondent key/value pair in the distributed object caching service.

On the other side, when a command issued by a client application will be received by the CU, the *command* module will be invoked for executing it. Parameters passed to the command's execution plug-in module will specify the action to be taken according to the instructions provided. The use of separated threads assures that requested periodicity of dataset refreshing is preserved even during commands' execution while serialization of parameters passed to the command plug-in allow a common interface for all the commands to be implemented.

Live Data Caching, DAQ and Central Services

Caching of live data, by means of distributed object caching service, and continuous archiving of accelerator data, by using a distributed key/value database, are the main innovations introduced by the !CHAOS paradigm.

DOC service is distributed over many nodes working together to provide clients with a single virtual pool of solid-state memory by sharing a portion of the RAM of each node. Objects are stored in memory as *key/value* pairs and a given object is always stored and always retrieved from the same node in the cluster, unless the number of node changes for any reason.

In !CHAOS a *key* identifies a unique dataset of the control system that is the set of information used to fully describe a real, or virtual, accelerator device.

Each dataset is periodically refreshed by the Control Unit in charge for the corresponding device. In the DOC service, dataset refreshing means overwriting the old data with newer describing the actual state of the device. Refresh rate is set and adjusted independently for each device and typical values span from milliseconds to few seconds.

Similarly, for the DAQ, key/value pairs are pushed to a node of the distributed KVDB to be stored on disks. In this case the *key* contains encoded both the unique dataset indicator and the timestamp. By querying the DAQ for all the datasets corresponding to a given timestamp the status of the accelerator at that particular time can be recovered.

The software opted for implementing the DOC and the KVDB are memcached[4] and mongodb[5] respectively. They demonstrated to offer the needed features and performances and are supported by a large and growing community of users. Nevertheless the abstraction of services provided by the !CHAOS components would allow their replacement, with other implementation of DOC and KVDB, without any modification of both its functionalities and API.

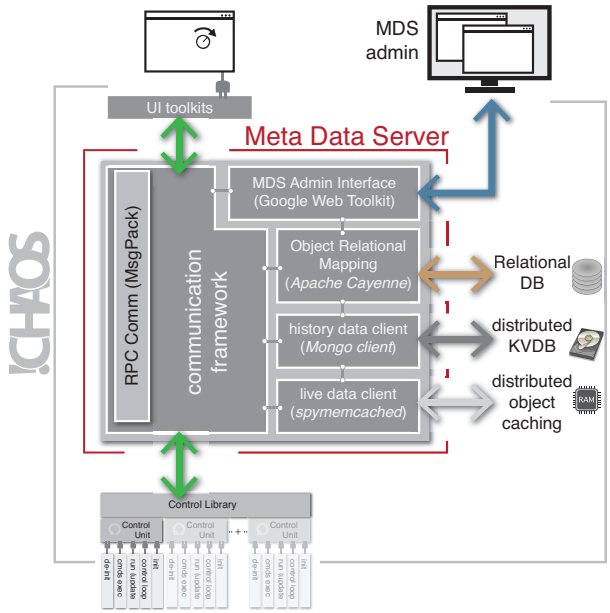


Figure 3: The !CHAOS meta-data server.

A key aspect in the !CHAOS development is the solution used to format data for both storing it, either into DOC or KVDB, and passing it between the different CS components.

Binary serialization is a convenient solution for flattening even complex data structure into a one-dimensional stream of bits suited for transmission through network. It is well suited especially for large binary arrays that are frequently included in datasets of accelerator's components.

What's more, both DOC and KVDB allows binary serialized data. In !CHAOS BSON[6] serialization is used for encoding dataset to be stored both in the *live data* DOC and in the DAQ . BSON serialization is also used by UI toolkit (see next section) for formatting commands sent to front-end Control Units and for passing parameters between CU and device management plug-ins.

Another fundamental component in the !CHAOS framework is the meta-data server (Fig.3). It is designed to store information such as CU configuration, commands list, commands and data semantic, naming service etc.

Object Relational Mapping (ORM) packages will be used to abstract the relational database, used for storing meta-data, by mapping its tables into Java object.

User Interface Toolkit

Client access to !CHAOS services will uniquely allowed through the API provided by the *User Interface toolkit*.

The set of API aiming to abstract and simplify the access of client applications to the !CHAOS service.

Fig.4 shows the logical structure of the UI toolkit layer with the blocks of API to client application and the substrate of API for the abstraction of the !CHAOS services.

In the figure is also introduced the concept of *UI data*

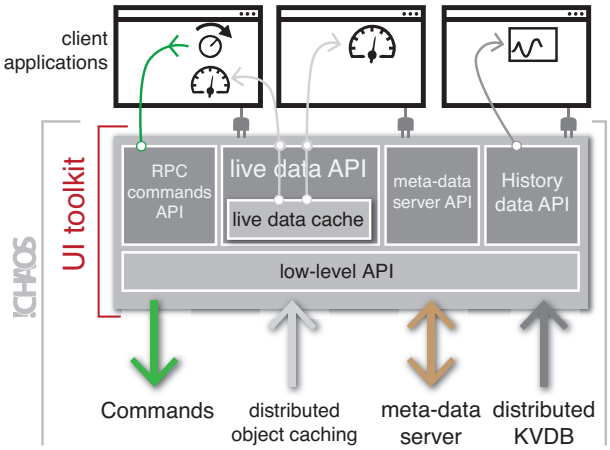


Figure 4: The User Interface toolkit components.

cache we are currently developing to achieve a further improvement of UI toolkit performance.

Practically it consist of a local cache of data and meta-data where UI toolkit APIs may store and share both frequently used meta-data, produced by queries to MDS, and live data read from distributed object cache. Similarly to distributed live data caching, we are considering a solution based on a key/value object caching to store locally this information. Caching of live data will take into account the refresh rate of the particular device dataset for setting its expiration time.

While most of the !CHAOS code is written in C, C++ and Java, development of both client applications and device management plug-ins should include a larger selection of programming languages.

Since at INFN LNF and Roma TV there is a long tradition in developing control and data acquisition systems with National Instruments LabVIEW we already started remodeling of existing front-end software to adapt it to !CHAOS DMP requirements.

On the client side, UI toolkit will provide APIs for most common measurement and analysis software like Matlab and the before mentioned LabVIEW.

CONCLUSION

!CHAOS is a scalable control system infrastructure providing all the services needed for communication, data archiving, timing, etc. in a control system for a particle accelerator or any other large apparatus. Front-end controllers and GUI applications can be seen as plug-ins that access and expand its functionalities.

The innovative communication framework is based on a distributed object caching service while continuous archiving of data is implemented by means of a non-relational distributed key/value database.

The use of the before mentioned software technologies introduces a new paradigm of control system in which the two layers representing the front-end and the client level are complemented by a third intermediate level collecting and distributing the the data produced by the lower front-end controllers.

The control groups at INFN-LNF and INFN-Roma Tor Vergata are committed to finalizing the development of this conceptual design, validating its functionalities and performance, and candidate !CHAOS as the control system for future INFN particle accelerators.

REFERENCES

[1] SuperB-CDR2 INFN-LNF-11/9(P) 15 Jun 2011

[2] G. Mazzitelli *et.al.*, "High Performance Web Applications for Particle Accelerator Control Systems", Proceedings of IPAC2011, San Sebastian, Spain, pp.2322-2324, http://www.JACoW.org.

[3] M.E. Thuot, L.R. Dalesio, "Control system architecture: the standard and nonstandard models," Particle Accelerator Conference, 1993., Proceedings of the 1993, pp.1806-1810 vol.3, 17-20 May 1993

[4] http://memcached.org.

[5] http://www.mongodb.org.

[6] http://bsonspec.org.

INTEGRATING A WORKFLOW ENGINE WITHIN A COMMERCIAL SCADA TO BUILD END USERS APPLICATIONS IN A SCIENTIFIC ENVIRONMENT

M. Ounsy, S. Pierre-Joseph Zephir, K. Saintin, G. Abeille, Synchrotron Soleil, Gif sur Yvette, France
E. de Ley, Isencia, Gand, Belgium.

Abstract

To build integrated high-level applications, SOLEIL is using an original component-oriented approach based on GlobalSCREEN, an industrial Java SCADA [1]. The aim of this integrated development environment is to give SOLEIL's scientific and technical staff a way to develop GUI applications for external users of beamlines. These GUI applications must address the two following needs: monitoring and supervision of a control system and development and execution of automated processes (as beamline alignment, data collection and on-line data analysis).

The first need is now completely answered through a rich set of Java graphical components based on the COMETE [2] library and providing a high level of service for data logging, scanning and so on. To reach the same quality of service for process automation, a big effort has been made for more seamless integration of PASSERELLE [3], a workflow engine with dedicated user-friendly interfaces for end users, packaged as JavaBeans in GlobalSCREEN components library.

Starting with brief descriptions of software architecture of the PASSERELLE and GlobalSCREEN environments, we will then present the overall system integration design as well as the current status of deployment on SOLEIL beamlines.

INTRODUCTION

GlobalSCREEN is a SCADA supporting the JavaBeans technology. Java beans are reusable graphical unitary components often used in Java GUI software projects (Swing components). GlobalSCREEN provides a user-friendly GUI to implement Java applications by drag-dropping. It also proposes to integrate all developed reusable components in a shared library.

Passerelle is a toolkit provided by Isencia [4] for designing sequences (and more generally data workflows) in a "drag and drop" graphical environment. Its core functionalities are based on the Java technology standards.

To design/develop/program sequences, called models, ISencia provides a graphical IDE (Integrated Development Environment). The IDE offers an execution engine as well as a number of essential framework services.

To use Passerelle models in a non-development environment, another Passerelle execution environment is provided by a GUI (Graphical User Interface) called PasserelleHMI. This GUI is able to display a panel on top of any model, giving access to all its configurable parameters.

This GUI component was the principal tool for the scientific acquisition workflow integration within GlobalSCREEN. That solution was not enough flexible to provide GlobalCREEN application developers with the ability to devise synthetic views where component addressing control/supervision/monitoring can be inter-mixed with elementary Passerelle sequence parameter configurations. There was also the need, behind the scenes, for different sequences to be triggered depending on the different user interaction scenarios with the GlobalSCREEN application.

All these consideration motivated the modification of the overall system integration design.

THE INTEGRATION DESIGN

The basic idea of the system integration design is to use, for the Passerelle environment, the same philosophy used for Tango integration in GlobalSCREEN through the COMETE library and its use of the Tango client API.

Hence the library of graphical components named Passerelle Widgets have been extended to give access to all the Workflow engine features through the use of PasserelleAPI module.

In this new integration design we also took care of the possibility to launch the final end user applications locally or remotely taking benefit from the compatibility of GLOBALSCREEN is compatible with the Java Web Start technology. Java Web Start is a framework developed by Sun Microsystems that allows users to start application software for the Java Platform directly from the Internet using a web browser. This enables SOLEIL users to remotely access the same rich client platform applications they use locally, in daily operation.

To achieve this, all the API Interface libraries needed for client access to each of the services are available with two different implementations, one local and one remote.

For the remote implementation:

- COMETE was adapted to access the control system via an intermediate layer. This layer translates Tango requests to the TangORB API in the local case or to HTTP requests reaching a WebTangORB server for remote access. The WebTangORB server then translates these requests to TangORB API calls to the underlying control system.
- The PasserelleAPI Interface is implemented as HTTP requests addressing a remote server complying with the W3C Webservice standard for

software systems designed to support interoperable machine-to-machine interaction over a network.

APPLICATION EXAMPLE

We will here describe how the use of this new design of Passerelle Workflow engine integration within GlobalSCREEN helped the ANTARES beamline at SOLEIL to provide their users with an integrated graphical user interface with all the needed features to handle their acquisition process.

This beamline (see Fig. 1) was designated for the complete determination of the electronic structure of condensed matter, using the angle resolved photoemission spectroscopy and the photoelectron diffraction (PED), which allows a local characterization of the geometric structure.

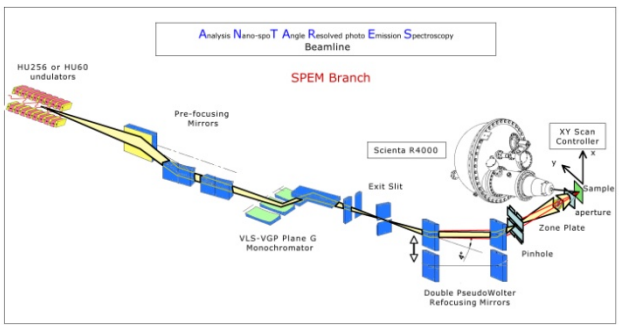


Figure 1: Beamline design.

As shown in the GlobalSCREEN Welcome page (see Fig. 2), the user can access to all the steps needed to drive his experiment: beamline control and supervision, alignment, homing of optical devices, running experiments.

Figure 3 displays the view giving access to all the entry points to all the optics alignments needed before starting the acquisition.

Figure 2: Application welcome page.

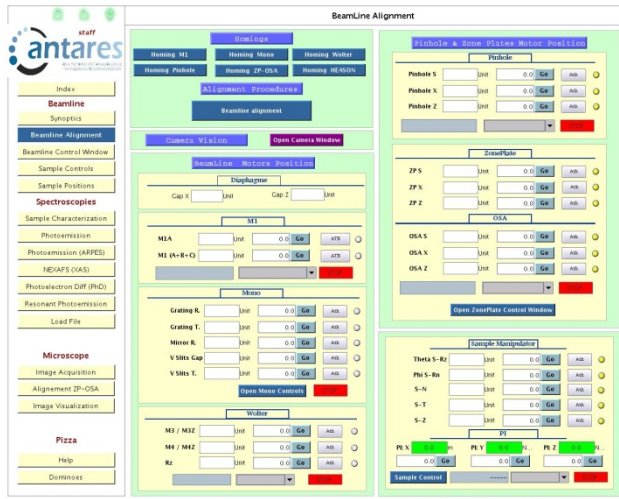


Figure 3: Alignment workflows.

In the Figure 4 below we can see the panel that pops up when we launch the monochromator homing button.

All the buttons entitled “Grating and Mirror”, “Vertical Slits”, “Horizontal Slits” are in fact unitary elementary Passerelle Widgets connected to predefined workflows that can fire Passerelle sequences to be started, resumed or stopped.

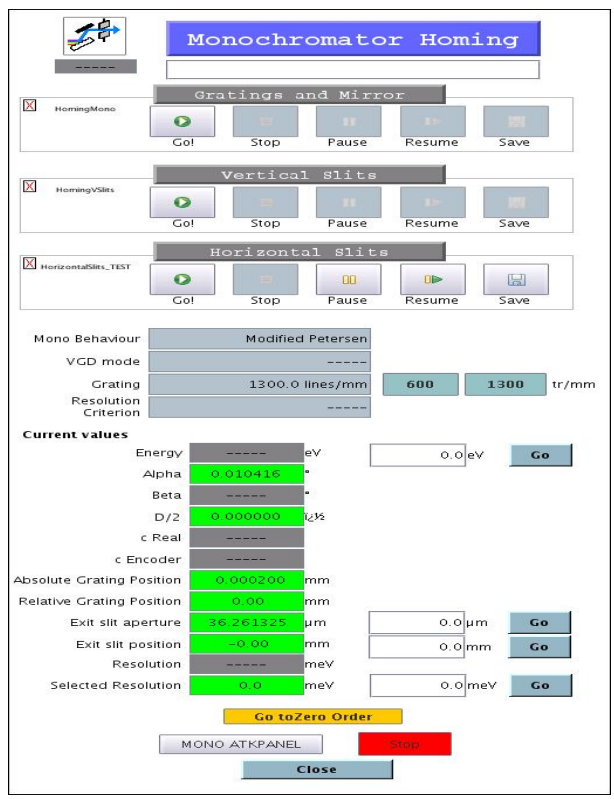


Figure 4: Monochromator alignment.

Figure 5 shows the GUI for performing a photoemission experiment using a Scienta detector. Here again, Passerelle Widgets for the acquisition workflow parameterization, sequence launching are seamlessly intermixed with Comete Widgets for equipments control and data acquisition monitoring (Scienta spectrum and image) giving the user total control of their experimental operation in the same application and hiding all the software layers operating behind the scenes.

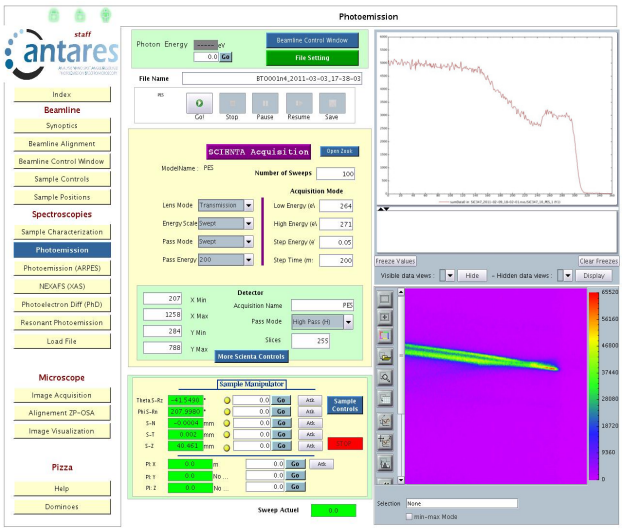


Figure 5: Scienta experiment view.

STATUS OF DEPLOYMENT

The figure below shows that using the Passerelle Workflow engine as a strategic tool for performing experimental acquisition processes in an integrated manner when combined with GlobalScreen is now becoming a reality at SOLEIL. Even if some beamlines are yet to be convinced of the benefit they can obtain from this way of operating beamline, the success obtained on doing it on many beamlines will facilitate this in a near future.

Beamline	Used by beamline stuff	Used in daily operation	Convinced by The product	To be convinced
ANTARES				
SMIS				
CASSIOPEE			0	1
CRISTAL	1	1	0,5	0,5
DEIMOS	1	1	1	
DESIRS				
ODE	1	1	1	
PROXIMA1	1	1	1	
SAMBA				
SWING	1	1	1	
TEMPO	1	1	1	
METROLOGIE			0	1
ANTARES	1	1	1	0
PLEIADES	0,5		0	1
LUCIA			0	1
MARS	1	1	1	0
SIXS	1	1	0,5	1
DISCO			0	1
DEIMOS			0	1
MICROFOCUS			0	1

Figure 6: Passerelle Deployment status.

REFERENCES

[1] V. Hardion, M. Ounsy, K. Saintin "How to Use a SCADA for High-Level Application Development on a Large-Scale Basis in a Scientific Environment", ICALEPS 2007

[2] G.Viguier, K.Saintin https://comete.svn.sourceforge.net/svnroot/comete, ICALEPS'11, MOPKN016.

[3] A. Buteau, M. Ounsy, G. Abeille "A Graphical Sequencer for SOLEIL Beamline Acquisitions", ICALEPS'07, Knoxville, Tennessee - USA, Oct 2007.

[4] Isencia http://www.isencia.be/

A GENERAL DEVICE DRIVER SIMULATOR TO HELP COMPARE REAL TIME CONTROL SYSTEMS

M.Mohan, European Gravitational Observatory, Cascina (Pisa) , Italy.

Abstract

***S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition systems (**SCADA**) such as *Epics*, *Tango* and *Tine* usually provide small example device driver programs for testing or to help users get started, however they differ between systems making it hard to compare the SCADA. To address this, a small simulator driver was created which emulates signals and errors similar to those received from a hardware device. The simulator driver can return from one to four signals: a ramp signal, a large alarm ramp signal, an error signal and a timeout. The different signals or errors are selected using the associated software device number. The simulator driver performs similar functions to *Epic's* clockApp [1], *Tango's* TangoTest and the *Tine's* sinegenerator but the signals are independent of the SCADA. A command line application, an *Epics* server (IOC), a *Tango* device server, and a *Tine* server (FEC) were created and linked with the simulator driver. In each case the software device numbers were equated to a dummy device. Using the servers it was possible to compare how each SCADA behaved against the same repeatable signals. In addition to comparing and testing the SCADA the finished servers proved useful as templates for real hardware device drivers.

INTRODUCTION

The current vacuum system at EGO [2] is controlled by os9 crates and uses custom software. The software is used to monitor values such as temperature, pressure and displacement. This control system is now being updated for Advanced Virgo. The os9 crates will be replaced by PLCs and a new SCADA will replace the custom software. The SCADA chosen may also be used for other systems at EGO.

Integration of new devices into a SCADA requires the hardware be setup correctly and the protocol (such as modbus) be implemented correctly. Rather than setup a PLC and test different conditions a small library was written to simulate signals. The simulator library generates simple scalar signals and errors similar to those which could be generated by a PLC.

Epics, Tango and Tine SCADA

Epics, tango and tine SCADA were chosen for comparison as they are all stable mature systems which are actively supported and developed. They are all open source projects which run on multiple platforms. The preferred platform at EGO is linux and both epics and tango have been recently packaged for Debian linux using the aptitude [3] package manager which eases installation and updating.

Epics is the most popular and widely supported SCADA followed by tango and then tine. Larger support implies stability [4] and a larger supply of pre-written device servers whereas smaller projects tend to be more flexible and use more advanced techniques.

The architectures of the SCADA investigated are similar and a generalised SCADA view is illustrated in Figure 1. Information from devices such as PLCs D1, D2, D3, and D4 is packaged and put on "software bus" by a Device Server. Different tools under the control of an operator are used to monitor and control the devices. Simulator signals were used in place of signals from D1, D2, D3 and D4.

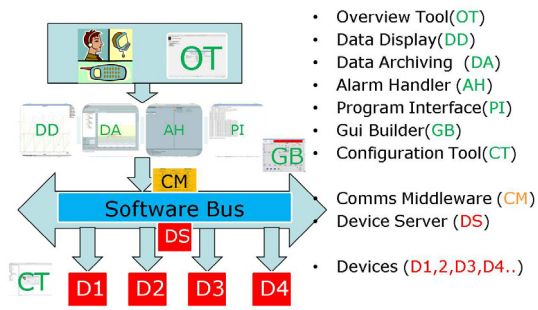


Figure 1: SCADA requirements.

SIMULATOR DEVICE SIGNALS

Four generated signals which are named according to the type of signal returned are shown in Table 1.

Table 1: Simulator Device Names

Device Number	Device Name
1	test/simulator/ramp
2	test/simulator/rampx2
3 (default)	test/simulator/error
4	test/simulator/timeout

The test signal is selected using the Device Number in the same way a modbus value is read using its modbus address. The signals are…

1. ramp: A ramp which rises from 0 to 59.999999 every minute.
2. rampx2: A large ramp which rises from 0 to 119.999998 every minute. Used to test alarms.

3. error: An error signal set at 60. Used to test how the SCADA reacts if the PLC returns a fault (-1).
4. timeout: A timeout signal. Used to emulate when there is no response from a PLC.

The signals are shown in figures 2, 3 and 4 using epics, tango and tine data displays.

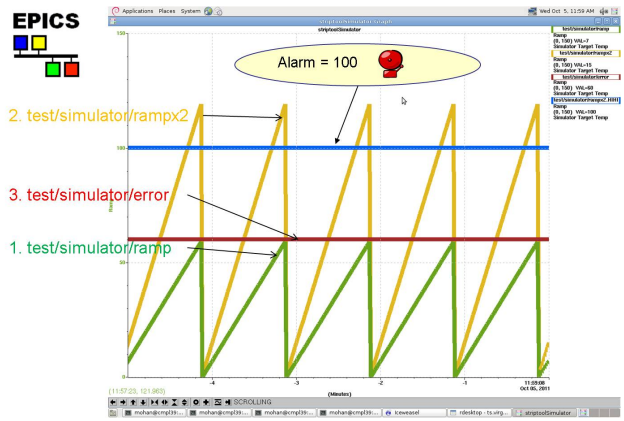


Figure 2: Simulator Signals using epics striptool.

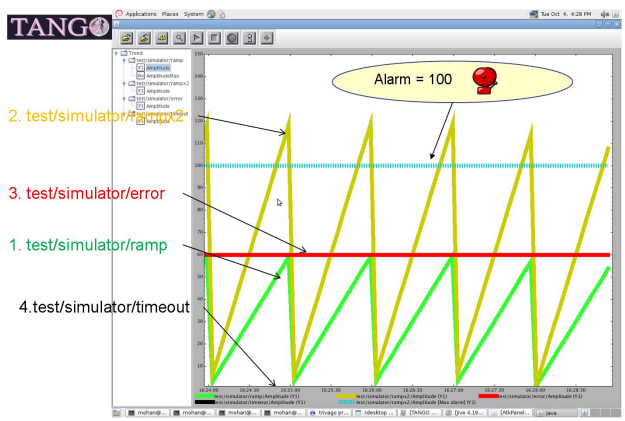


Figure 3: Simulator Signals using tango atkmoni.

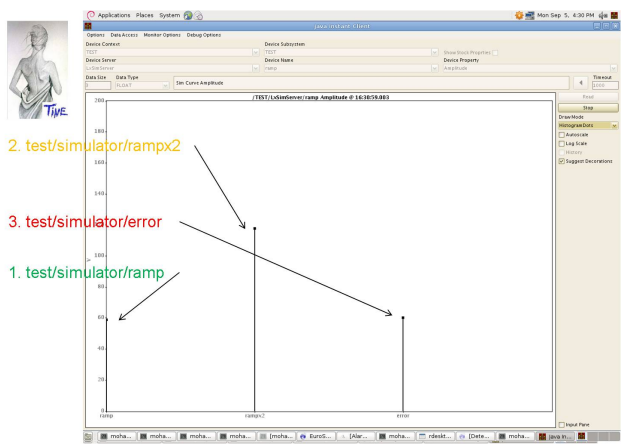


Figure 4: Simulator Signals using tine JClient. "Histogram view". There are several view types in Tine.

How Simulator Signals are Generated

A small c program was used to generate the four signals shown in Figure 2, 3 and 4. The function to generate the signals is shown in Figure 5.

```
…
int GetAmplitude(int DeviceNumber, double *Amplitude)
{
      struct timeval  tv; struct timezone tz;
      struct tm      *tm;
      gettimeofday(&tv, &tz); tm = localtime(&tv.tv_sec);

      // *Amplitude= Seconds time of system clock
      *Amplitude=((double)((1000000*tm-
>tm_sec)+tv.tv_usec)/1000000); //  0-59.999999

      int ErrorValue =0; // default 0=ok

      switch (DeviceNumber)
      {
            case 1: // 0-59.999999 = ramp
                  break;
            case 2: // 0-119.999998 = rampx2
                  *Amplitude=*Amplitude*2;
                  break;
            case 4: // =timeout
                  sleep(99999999);
                  break;
            default: // =error
                  *Amplitude=60;
                  ErrorValue=-1; //  -1 = Error
                  break;
      }
      return ErrorValue; // return 0 or -1
}
…
```

Figure 5: C function to generate Simulator signals.

A library called "libSimulator" was made using the code in Figure 5 and linked with SCADA device servers.

CREATING DEVICE SERVERS

What is a Device?

The concept of the device is at the heart of epics, tango and tine. A device can be considered a container of hardware data and an analogy for this is a shipping container because the devices all look the same although the contents are different. Like devices shipping containers also have unique names [3].

The "device container" in Figure 6 is labelled with static information such as the name of the device, units of measurement and alarm limits. The device is packaged by the device server which updates dynamic information in the device. When the device server is running the device is updated with dynamic content such as Amplitude, state and timestamp. This "device container" is then put on its

respective software bus (epics, tango or tine) where it is identified by its unique name.

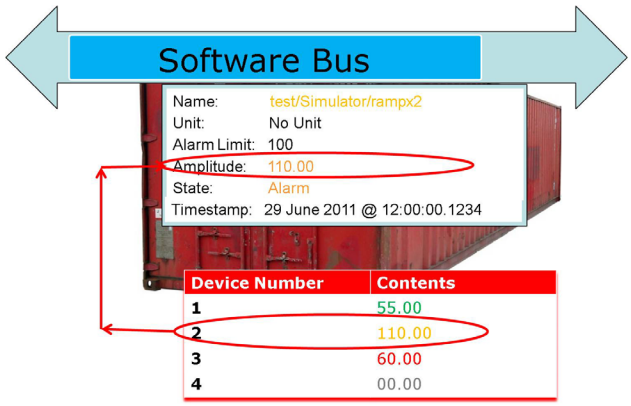


Figure 6: Example: Device test/simulator/rampx2 is updated periodically and results put on software bus.

Integrating Devices in Device Servers

The source code for the software device servers were generated using the template tools makeBaseApp, pogo and srvwizard which are supplied by epics, tango and tine respectively.

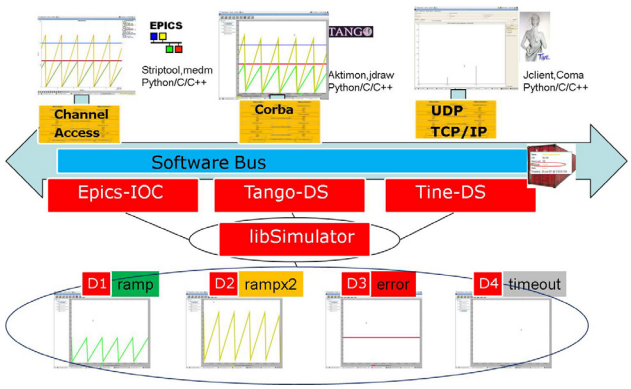


Figure 7: libSimulator test setup.

The device servers were linked to the libSimulator library Figure 7 and device signals generated by calling the function GetAmplitude Figure 5. The code added to device servers to read the devices is similar to the following.

C Command line

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
printf("Error\n");
```

Epics

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
recGblSetSevr(pai,STATE_ALARM,MAJOR_ALARM);
```

Tango

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
set_state(Tango::FAULT);
```

Tine

```
if(GetAmplitude(DeviceNumber,&Amplitude)==-1)
SetAlarm(SIMEQM_TAG,devnr,512,NULL);
```

The codes above read the Amplitude and sets an alarm if an ErrorValue (-1) is returned. This means the SCADA always set an error alarm for device 3 (test/simulator/error).

The default code generated for epics and tine hung on the timeout from device 4 (test/simulator/timeout) because the device server blocks waiting for a reply. Tango however by default sets an alarm state "timeout" and continued to read the other devices if no reply came within 3000 ms. There are several methods to overcome the timeout in epics and tine such as defining each device as thread or using epics async but this involves modifying the default generated code.

For real devices pre-written device servers can be downloaded from the SCADA websites and some hardware devices even come with inbuilt device servers such as the F3RP61 PLC from yokogawa ref [6] which includes an inbuilt epics IOC.

CONFIGURING DEVICES

SCADA configuration tools are used to input static information for devices such as names and alarm thresholds. For the simulator the Device number 1, 2, 3 or 4 must be associated with a unique device name. Epics naming is not compulsory and tango and tine use similar formats so it was possible to use the same Simulator signals names in all SCADA see Table1.

Epics

Configuration information for epics devices are stored in ASCII ".db" files and modified using a text editor. This device information is called a record. In the example in Figure 8 the field HIHI is used to set max alarm threshold to 100.

```
record(ai, "test/simulator/rampx2")
{
      field(DESC, "Signal ramp 0 to 119.999998 per
minute")
      field(INP,"2")
      field(HIHI, "100.0")
..
}
..
```

Figure 8: Epics configuration using .db file

Tango

Configuration information for tango devices are stored in a Mysql database and configured using the tango tool

jive. In the example in Figure 9 Max alarm for "test/simulator/rampx2" is set to 100 for the Amplitude attribute.

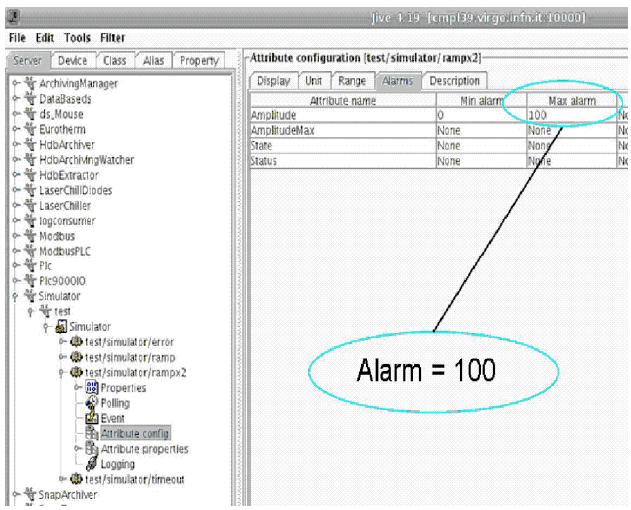


Figure 9: Tango configuration using jive.

Tine

Configuration information for tango devices are stored in a csv files and configured using spreadsheet tools or a text editor. Tine also provides the alternative options of storing configuration information in xml files or in the code.

In the example in Figure 10 High alarm for "test/simulator/rampx2" devices is set to 100.

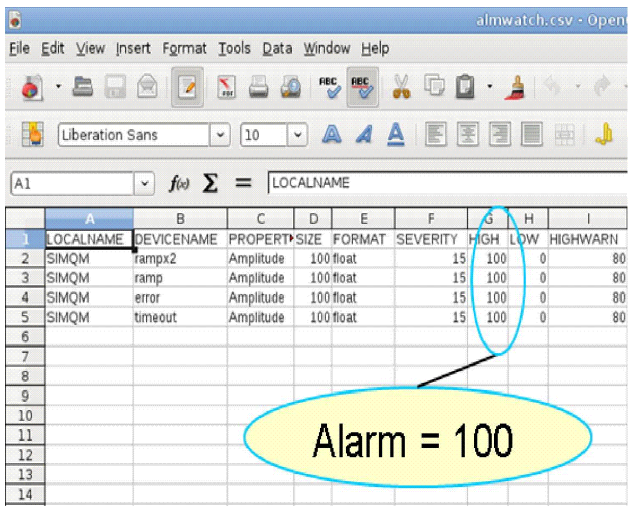


Figure 10: Tine configuration using an excel tool.

In the above configuration cases setting the alarm to 100 successfully caused the SCADA's Alarm handlers (or state handler for tango) to trigger a warning alarm when the property Amplitude exceeded 100 which happened for device rampx2 every 50 seconds.

SCADA TOOLS

After device information had been packaged and configured correctly they are put on the respective "software bus" and made available for the epics, tango or tine tools.

Some tools to view data online in epics, tango and tine are shown in figures 2, 3 and 4. These data display tools are called striptool, atkmoni and JClient respectively.

There are many other monitoring and control tools for epics, tango and tine which perform functions such as Data Display, Alarm Handling, Data Archiving and gui Building (Figure 1). The tools have many similarities and in fact the gui builder tool jddd [7] can be used to build guis for epics, tango and tine. In all cases a unique device name is used to identify the device. Additional tool comparisons can be found in reference [8].

CONCLUSION

Epics, tango and tine are all able to fulfil the requirements for the vacuum control at EGO and more. The learning curve for writing and configuring device servers is high but once mastered devices can be integrated quite quickly. There are many pre-written device servers available especially for the more popular SCADA. These pre-written device servers can be freely downloaded although they must still be configured and tested.

The Simulator is a simple aid to generate repeatable signals independently of the SCADA or hardware. It was useful for learning to write simple device servers for epics, tango and tine and for comparing SCADA tools using known signals.

All the SCADA provide useful tools for monitoring and controlling devices Figure 1. The tools are similar and some tools such as jddd [7] can even be used with all epics, tango and tine.

REFERENCES

[1] F.Furukawa,"Very Simple Example of EPICS Device Suport", http://www-linac.kek.jp/epics/second
[2] http://www.ego-gw.it/public/about/whatIs.aspx
[3] http://www.isbu-info.org/all_about_shipping_containers.html
[4] Raymond, Eric S.,"The Cathedral and the Bazaar", October1999, http://www.catb.org/~esr/writings/cathedral-bazaar/
[5] http://en.wikipedia.org/wiki/Intermodal_container
[6] A. Uchiyama… Proceedings of PCaPAC08, Ljubljana, Slovenia, Development of embedded epics on F3RP61-2L.
[7] http://jddd.desy.de/
[8] M.Mohan, VIR-0371A-11.pptx, "Advanced Virgo Vacuum control Epics, Tango and Tine SCADA comparison", https://tds.ego-gw.it/ql/?c=8426

A UML PROFILE FOR CODE GENERATION OF COMPONENT BASED DISTRIBUTED SYSTEMS

G. Chiozzi, R. Karban, L. Andolfato, ESO, Garching Germany
A. Tejeda, Universidad Católica del Norte, Antofagasta, Chile

Abstract

A consistent and unambiguous implementation of code generation (model to text transformation) from UML must rely on a well defined UML profile, customizing UML for a particular application domain. Such a profile must have a solid foundation in a formally correct ontology, formalizing the concepts and their relations in the specific domain, in order to avoid a maze or set of wildly created stereotypes. The paper describes a generic profile for the code generation of component based distributed systems for control applications, the process to distil the ontology and define the profile, and the strategy followed to implement the code generator. The main steps that take place iteratively include: defining the terms and relations with an ontology, mapping the ontology to the appropriate UML meta-classes, testing the profile by creating modelling examples, and generating the code.

INTRODUCTION

In every phase of a new project, one recurring activity is the assessment of artefacts of previous projects for opportunities of re-use. This might include requirements, architectural principles, design and, eventually, code.

Unfortunately, it becomes often apparent that the documentation is poorly articulated or outdated. The knowledge acquired during implementation and commissioning is just reflected in the code and can be "rediscovered" only by reverse engineering.

The developed code is often tightly linked to a specific infrastructure, which might have become obsolete or cannot be adopted due to project constraints.

To overcome some of those deficiencies, platform dependent information needs to be segregated from the domain specific application, and yet it must be possible to keep the two parts aligned automatically.

This principle can be put into practice, using technology which has become recently mature and, robust, such as Model Driven Development [1].

We want to use this type of development at ESO for new projects and the upgrade of existing systems.

MODEL DRIVEN DEVELOPMENT

The availability of open standards like UML, MOF, CWM (www.omg.org/mda) and open tools, frameworks and transformation languages (www.eclipse.org/emf) is a sign that Model Driven Development technologies are becoming mature and widely adopted [2].

In this approach, the system to be developed is described using a precisely defined domain specific modelling language. The artefacts of the development activity (like documentation and code) are derived from the original model using model to model or model to text transformations. The process can consist of multiple layers, each of them moving from an abstract to a more concrete representation of the system or from domain specific to general purpose languages. At the end of the process, the actual application source code is generated.

Up to a certain level of detail, graphical notations are best suited for system modelling [3]. We have therefore adopted the Unified Modelling Language (UML - www.omg.org/uml) as the general purpose modelling language. By now, UML is accepted by engineers in several fields, also outside the software domain.

Moreover, there are tools available, supporting Model Driven Development based on UML, providing means to define modelling languages and model transformations starting from UML.

It is important to notice that this is a step forward with respect to *Model Based Development*, where models are used just in the early phases of a project and afterwards the code is written by hand. *Model Based Development* suffers from the non formal relation between model and code, which causes obsolescence of the models and misalignment with the code.

THE PROCESS

In order to be able to formally define and implement the Model Transformations that allow going from the model to the final code, it is necessary to have a non-ambiguous description of the specific domain concepts.

UML provides a powerful mechanism to express domain specific concepts: the *Profile*.

A *UML Profile* is a collection of definitions for *stereotypes*, *tags* and *constraints* that customize UML for a domain, redefining the semantics of the modelling language in an additive way. This means that, inside a model, different but compatible UML profiles can be used at the same time to specify different aspects of the system.

We decided to develop a UML Profile to capture the meta-model necessary to describe our typical architecture of a telescope and of its instrumentation.

We started with a comparative analysis of the requirements and architecture of projects developed in the last years. From this analysis a first version of the profile was produced.

Unfortunately, while building the profile we realized that there were very limited possibilities to verify automatically the correctness and un-ambiguity of our models with respect to the meta-model. As a matter of fact the only type of validation allowed is based on the definition and verification of UML constraints, which are still not properly supported by most UML tools.

A proper level of formality can be obtained by defining an *ontology* [4] for the domain, i.e. a formal description of the concepts and of their properties, features, attributes, restrictions and relations.

We created an ontology using UML class diagrams, starting from the ones we used to identify the profile elements (Figure 1). However, those concepts and their relations should be expressed in an *ontology description language* like OWL [4].

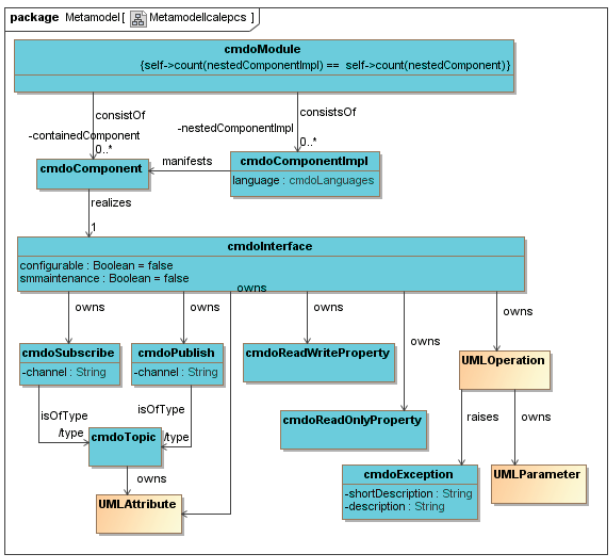


Figure 1: Snippet of the ontology as UML metamodel.

The resulting ontology is heavily affected by our experience and our specific focus, therefore groups with different background might create a different set of concepts to model the same domain.

Depending on the sophistication of the model transformations and of the tools used, we can get different approximations to the real final application.

We have adopted an iterative process consisting of the following steps:

- Analyze and compare parts of real systems we have implemented
- Describe in the *ontology* the common concepts
- Map the ontology to a profile
- Model the system with the new elements
- Implement the transformations
- Compare the generated code with the original code: it must be readable and reliable and it must be possible to add easily hand-crafted code
- Analyze what needs to be modelled and/or refactored at the next iteration

In this work we apply a few basic practical rules:

- An element of the ontology should be mapped in the profile only if it is worthwhile to model, i.e. if it takes less effort to transform the model representation with respect to an handcrafted implementation. The gain should consider parameters like one-to-many transformation scenario, copy&paste errors, or the number of times the element appears in our models.
- If a concept of the ontology applies to all model elements, it should be implemented via an external parameter passed to the transformation engine, while if it applies only to some model elements, it should be a stereotype or a tag.
- If a concept of the ontology applies only to some of the supported platforms, the transformation is ignored for the other platforms.

In the beginning we must expect to add code by hand for substantial parts of the application on top of a simple auto-generated skeleton, but an iterative process of extensions will add more and more automation.

The selection of the transformation language plays an important role in the definition of the profile, because some languages may provide better support for some profile element than others (for example, trading off between using stereotypes or tags favours stereotypes when using the Xpand language). We selected EMF and the Xpand tool set [5] because already used at ESO for some other projects. More details on the code generation process and its implementation are provided in [6].

THE COMODO UML PROFILE

In the overall system model, which includes the representation of hardware components and is typically defined using the System Modelling Language (SysML) profile, our new COMODO (COmponent MODelling for Observatory) profile allows to and model the parts used to generate the code of the real control system.

By keeping the abstract and Platform Independent information separated from the concrete Platform Specific information, we can generate application code for the same system on different platforms by simply replacing the model-to-model transformation for this last layer. A significant effort was spent in separating the Platform Independent Model (PIM) from the Platform Specific Model (PSM).

We will describe here below and in Table 1 some profile elements.

Distributed Components

Telescope control systems fit very well in an architecture based on Distributed Components, i.e. in an architecture where independent entities (called Components) communicate concurrently and asynchronously with each other.

In order to keep a clear distinction between PIM and PSM, we identified three entities in our meta-model:

- **Interface (cmdoInterface - PIM)**
- **Component (cmdoComponent - PIM)**
  In particular, state charts are used to specify behaviour. As described with more details in [6], state charts have been successfully applied at ESO for modelling reactive systems. An extensive usage as a core element of design, supported by code generation, simplifies the development.
- **Component Implementation (cmdoComponentImpl - PSM)**

Table 1: Basic Elements of the Comodo Profile

Name	Base Classifier	Documentation
cmdo Interface	UML Interface	Part of: PIM Public interface that the Component is exposing. Can include: • Operations provided (synchronous invocation paradigm) • Published and subscribed data structures (asynchronous communication paradigm) • Properties, to represent data attributes that can be read/set/monitored • Error handling and exceptions specification • Signals to be used as triggers for the state machine
cmdo Component	UML Class	Part of: PIM Platform independent representation of a Component as it is going to be implemented, realizing the corresponding Interface and optionally specifying the dynamic behavior with a state chart. A design including algorithms or private attributes/operations used in any implementation can be specified here.
cmdo ComponentImpl	UML Artifact	Part of: PSM The Component Implementation is a manifestation of the corresponding cmdoComponent. It represents the actual implementation for a specific platform. If we have multiple implementations for the Component, for example with different programming languages or for different infrastructures, we can model each of them independently without affecting the platform independent representation.
cmdo Topic	UML Signal	Part of: PIM A Topic is a type for (published or subscribed) attributes or parameters. Used as triggers by State Machine.
cmdo Publish	UML Property	Part of: PIM Used to specify whether an attribute in the interface whose type has stereotype cmdoTopic has to be published.
cmdo Subscribe	UML Property	Part of: PIM Used to specify whether an attribute in the interface whose type has stereotype cmdoTopic has to be subscribed.
cmdo Module	UML Package	Part of: PSM This is an organizational unit for specifications of components, interfaces, etc. It identifies a set of deliverables.
cmdo Machine	UML Node	Part of: PSM Machines represent the physical system were one or more containers can run. The deployment of containers is modeled as attributes of machines.
Cmdo Exception	UML Signal	Part of: PIM Exception are used to model errors returned by the interfaces.
Cmdo Container	UML Exe.Env	Part of: PSM Containers are the environments where one or more component implementations can run. The deployment of component implementations is modeled as attributes of containers.

Deployment

The deployment information for the system is part of the PSM, since the same architectural Components can be deployed in different ways depending on the platform.

Moreover, we can have different deployments for the same applications in different phases of the project or for different purposes (testing, commissioning or operation). Deployment is typically a separate responsibility from development and is done by different actors (*system configurators* with respect to *developers*). Our meta-model assumes that Components are deployed in Containers, according to the Component/Container paradigm used by many distributed infrastructures.

A Container provides to Components a set of services (like logging or Component location) that helps shielding the implementation of Components from the specific software infrastructure adopted. In the simplest case of the VLTSW infrastructure the Container is just implemented with a CCS Environment [7], while in the case of ACS a Container is a process that can dynamically load and host multiple Components [8].

E-ELT PROTOTYPE INSTRUMENT

As an example, we show here the model corresponding to a small instrument described in [9], which was developed as a prototype for the evaluation of software and electronics for E-ELT instrumentation (Figure 2).

In this model we define generic interfaces for Subsystems and Devices (Figure 3) which are specialized for a particular application. For example, from the Device we specialize Motors and from that Filter Wheels.

All Devices share a state machine specified in the Device Component platform independent definition.

The model includes the implementation specification for all these components in Java and a test deployment.

With the existing code generator, we can produce from this model a skeleton implementation of this system for the ACS platform, providing sufficient functionality to start-up the system and test interfaces and state machines.

The developer can immediately implement the behaviour code (such as state machine actions and activities) inside the skeletons. With the help of an application framework, a limited knowledge of the

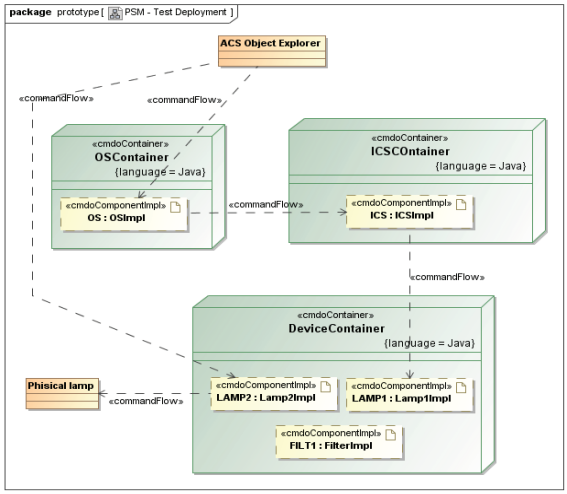


Figure 2: Partial deployment of the instrument.

specific platform is sufficient to complete the development, flattening considerably the learning curve.

In the case of changing platform, all of the PIM of the model would remain the same and probably (depending on the platform) also a big part of the PSM.

Definitions like the ones for Device or Motor are very generic and could therefore become part of the Profile itself or of a domain/project specific extension.

FUTURE DEVELOPMENT

Both, the ontology and the profile need to be extended to cover more domain (telescope and instrumentation) specific aspects of our systems. At the current stage, we have just a generic component/container distributed system model, but we have started to define an architecture framework for telescopes and instruments.

This will be specified through an ontology, with a proper ontology specification language, to allow formal checking.

On the code generation side, we are working on extending the support for ACS to the C++ language. This is also the first step to support the VLT platform, which is essentially based on C++.

CONCLUSIONS

This work is helping us significantly in reaching the objective of modelling our systems in a platform independent way, leaving big parts of the code production to code generators. In this way, the same model can be reused for different infrastructures and, therefore, for different projects and with a longer life span. At the same time, developers are shielded from the details of the specific platform.

This allows us to work now on the modelling of E-ELT TCS and instrumentation without knowing what infrastructure will be finally used, but testing our architecture on the ACS platform (or on the VLT once the transformations will have been implemented).

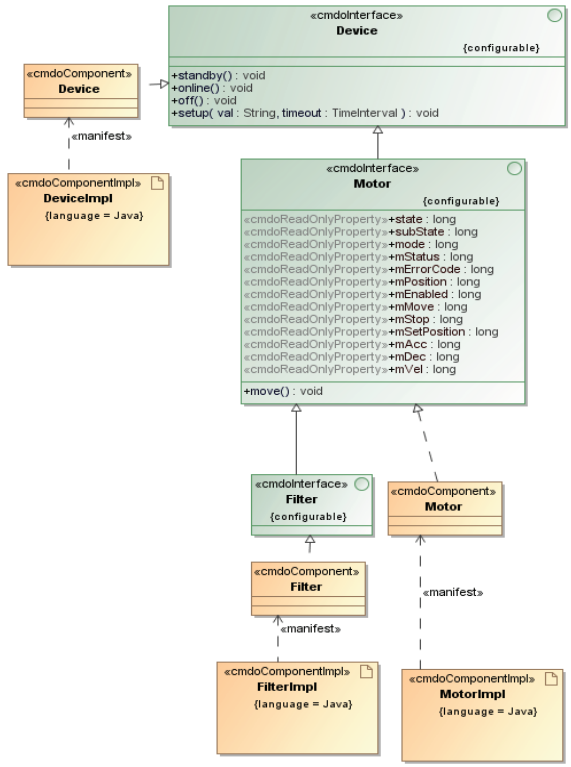


Figure 3: Specifications for motors in the prototype.

REFERENCES

[1] S. Staab et al., "Model Driven Engineering with Ontology Technologies", Reasoning Web: Semantic Technologies for SW Engineering (2010)

[2] G. Booch, et al., "An MDA Manifesto", MDA Journal, (2004)

[3] D. Harel, "Statecharts in the Making: A Personal Account", Communications of the ACM, 03/2009, Vol. 52, No.03

[4] N. F. Noy, D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology", Tech. Rep. SMI-2001-0880, 2001

[5] B. Klatt, "Xpand: A Closer Look at the model2text Transformation Language" 12th European Conference on Software Maintenance and Reengineering, (2008).

[6] L. Andolfato et al., "A Platform Independent Framework for Statecharts Code Generation", this conference, (2011).

[7] M.J. Kiekebusch et al., "Evolution of the VLT instrument control system toward industry standards", Proc. SPIE 7740, 77400T (2010)

[8] G. Chiozzi et al., "ALMA Common Software (ACS), status and development", in [Proceedings of ICALEPCS] (2009)

[9] P. Di Marcantonio et al., "Evaluation of Software and Electronics Technologies for the Control of the E-ELT Instruments: a Case Study", this conference (2011)

UNICOS CPC6: AUTOMATED CODE GENERATION FOR PROCESS CONTROL APPLICATIONS*

B. Fernández Adiego, E. Blanco Viñuela, I. Prieto Barreiro, CERN, Geneva, Switzerland

Abstract

The Continuous Process Control package (*CPC*) is one of the components of the CERN Unified Industrial Control System framework (*UNICOS*) [1]. As a part of this framework, UNICOS-CPC provides a well defined library of device types, a methodology and a set of tools to design and implement industrial control applications. The new CPC version uses the software factory UNICOS Application Builder (*UAB*) [2] to develop CPC applications. The CPC component is composed of several platform oriented plug-ins (PLCs and SCADA) describing the structure and the format of the generated code. It uses a resource package where both, the library of device types and the generated file syntax, are defined. The UAB core is the generic part of this software, it discovers and calls dynamically the different plug-ins and provides the required common services. In this paper the UNICOS CPC6 package is introduced. It is composed of several plug-ins: the Instance generator and the Logic generator for both, Siemens and Schneider PLCs, the SCADA generator (based on PVSS) and the CPC wizard as a dedicated plug-in created to provide the user a friendly GUI. A tool called UAB Bootstrap will manage the different UAB components, like CPC, and its dependencies with the resource packages. This tool guides the control system developer during the installation, update and execution of the UAB components.

INTRODUCTION

In this paper, we introduce the UAB CPC6 (*UNICOS Application Builder - Continuous Process Control*) component as a part of the UNICOS (*Unified Industrial Control System*) framework. UNICOS is a control system framework, developed at CERN (*European Organization for Nuclear Research*), designed to implement control system applications. This framework provides a methodology, an object library and a set of tools to generate the control code for these applications. A UNICOS component or package is a part of the UNICOS framework which uses the UNICOS methodology to create a specific type of applications, provides its own object library and uses the UNICOS generation tools to obtain the control code for these applications. Currently there are several UNICOS components, used for different accelerator systems, such as CPC, CIET (*Cryogenics Instrumentation Expert Tool*), QPS (*Quench Protection System*) and SURVEY (Control system for the magnets alignment) [3] (See Fig. 1).

The CPC component has been designed to develop industrial control system applications for continuous processes. It has been used for more than ten years in continuous processes like the LHC (*Large Hadron Collider*) cryogenics system, the gas systems of the LHC experiments, interlock system and lately cooling & ventilation and vacuum systems [4]. The CPC package creates applications focused in the two upper layers of a control system (Supervision and Control) and sets the communication between them automatically. To achieve this, the CPC objects Baselines have been developed at the SCADA (Supervisory Control And Data Acquisition) and the control levels. The control layer is PLC (Programmable Logic Controller) based, currently it has been developed for the Siemens S7 and Schneider platforms and for CoDeSys (Controller Development System) which makes it platform independent, while the SCADA layer has been developed for PVSS, nowadays called WinCC OA (WinCC Open Architecture).

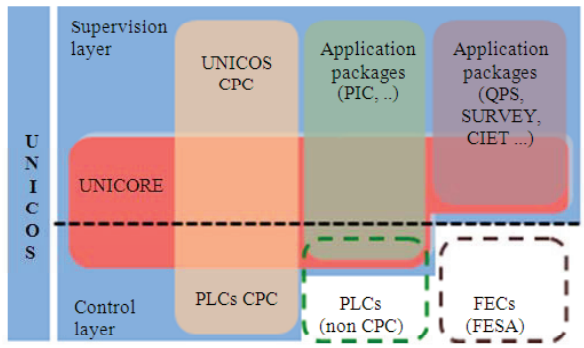


Figure 1: UNICOS Overview.

UNICOS CPC6

The UNICOS-CPC package presents three significant modifications from the previous version: the methodology, the objects and a new generation tool called UAB.

The UNICOS methodology has been improved providing a set of documents and some well-defined steps to develop CPC control applications. Following this methodology, the CPC user will be able to transform the process knowledge into a model based in a hierarchy of CPC objects (See Fig. 2), and implement the control system using that information. These objects have been improved

* Work partially funded by the Spanish Ministry of Science and Innovation

from the previous version, including new functionality and more flexibility for the users and operators [5].

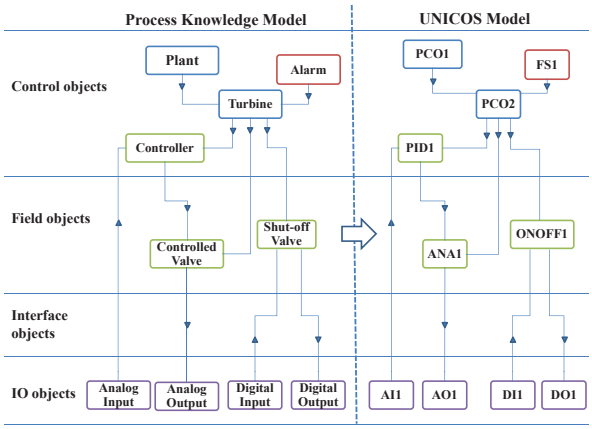


Figure 2: Example of CPC objects hierarchy.

This methodology contains the following steps: (See Fig. 3).

- Functional analysis: the process engineer transfers the process knowledge to the UNICOS Functional Analysis document.
- UNICOS Logic design: the process engineer and the control engineer work together to define the behavior of the process in the UNICOS language.
- Specification file filling: all the UNICOS objects are defined and parametrized using an Excel/XML file.
- Automatic generation: the instance and logic code for the PLC and the configuration file for WinCC OA are generated using the automatic generation tools.
- Logic completion: complete the automatic generation of the logic if necessary, following the UNICOS Logic design.
- Control code compilation.
- Application tests.
- Commissioning.
- Operation.

USING AUTOMATIC GENERATION TOOLS

The use of automatic generation tools allows development of control system applications with a high level of abstraction, decreasing the development, configuration and commissioning time and producing well-structured control code. For this new version, one of the most important requirements was to include a more flexible, extensible, consistent and user-friendly automatic generation tool. As result of these requirements the UAB has been developed. This tool generates the control code extracting information from the UNICOS objects definition and the process knowledge included in the specification file [2].

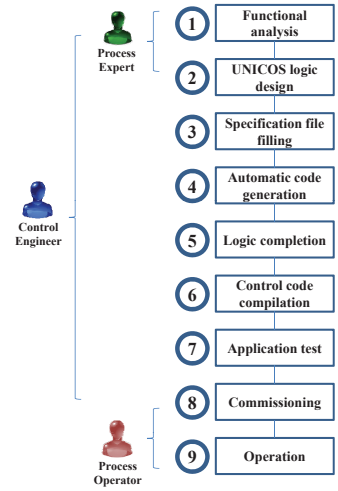


Figure 3: UNICOS-CPC6 methodology.

Architecture

The UAB architecture is composed of three main parts: the UAB core, the UAB plug-ins and the resource package (see Fig. 4). These three parts have different functionalities and life cycles.

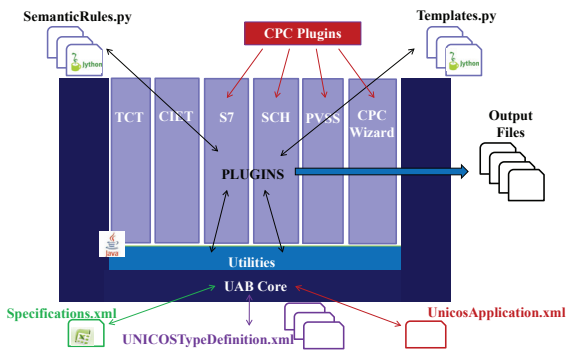


Figure 4: UAB Architecture.

The UAB has been developed in the Java language as well as other technologies like the Jython (Pyhton for Java) scripting language to define the syntax and the contents of the output files. The format for the configuration files is XML (*eXtended Markup Language*) and the technologies used to process these files are JAXB (*Java XML Binding*) and JXPATH (*XML Path Language for Java*). Apache Maven is the selected technology to build and manage the tool.

All the UNICOS objects, as the CPC objects, are defined as XML files (*DeviceTypeDefinitions*), where the

object structure is defined (Inputs, Outputs, Frond-End Parameters, SCADA Parameters, etc.). For this purpose, the so-called *UNICOS Metamodel* has been designed. This Metamodel is an XSD (*XML Schema Definition*) file used to describe and limit the contents of the DeviceTypeDefinitions [6].

UAB Core

The UAB core is the main and generic part of the tool. It is platform independent and provides the common services required by most of the plug-ins (for example User report, logging, etc.). It also has other functionalities, for example dynamically discovering the different plug-ins and asserting their validity, loading the UNICOS Project data information, connecting the plug-ins with the external files, etc.

UAB Plug-ins

A plug-in is the part of UAB where the structure and the format of the generated files are defined. It is platform dependent (i.e. Siemens, Schneider, etc.) and its main function is to generate these files.

Currently, the CPC package contains the following plug-ins:

- Siemens Instance Code Generator: it generates the instance and communication files in SCL (*Structured Control Language*) format and the symbols files containing the address mapping.
- Siemens Logic Code Generator: it generates the control logic files in SCL format.
- Schneider Instance Code Generator: it generates the instance and communication files in XML format.
- Schneider Logic Code Generator: it generates the logic files in XML format.
- WinCC OA Code Generator: it generates the configuration file for WinCC OA in TXT format
- CPC Wizard: it is a dedicated plug-in that provides a friendly GUI to drive the generation of CPC applications. It prompts the user for the mandatory data of the application, validates the user data and triggers the execution of the selected plug-ins with the specified parameters. The wizard's panels can be customized through an XML file and a set of predefined components (like text editors, radio buttons, etc.) (See Fig. 5).

These plug-ins use the services provided by the UAB core to perform the following tasks:

- Read the data from the input sources, like the UNICOS specification file or the DeviceTypeDefinitions.
- Process the semantic check rules to validate the input data.
- Execute the Jython scripts.
- Generate the output files.

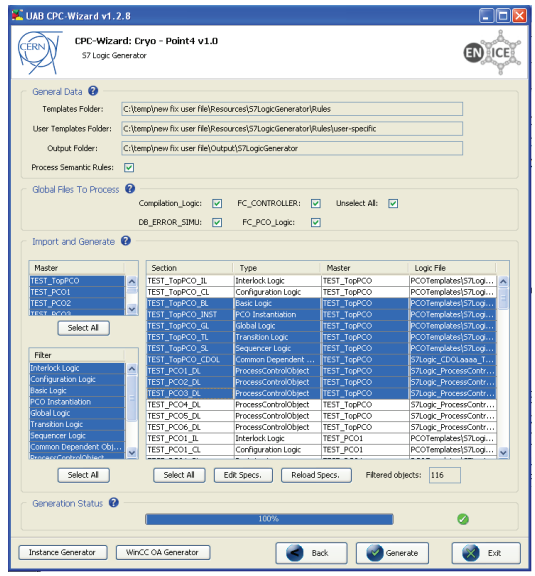


Figure 5: Wizard panel for the Siemens Logic generator.

The plug-ins are completely independent and it is possible to launch them through the wizard or one by one using the XML editor called FESA (*Front-End Software Architecture*) General editor from the FESA framework. Apart form the CPC plug-ins, more plug-ins have been developed in UAB. For example, for the CIET component the instance and WinCC OA configuration plug-ins have been created.

UAB Resources Package

The UAB resources package is the part of the tool where the CPC objects and the contents of the output files are defined, it also contains the semantic check rules used to validate the user inputs during the definition of the project. The CPC resources package contains the following list of elements:

- Generation Templates: these templates, written in Jython scripting language, define the syntax and the contents of the generated control code. There are five types of generation templates: the Siemens instance templates, the Siemens logic templates, the Schneider instance templates, the Schneider logic templates and the WinCC OA templates.
- DeviceTypeDefinitions: the XML files which contain the definition of the CPC objects.
- Baselines: the baseline is the definition of the CPC objects in PLC code and the additional common code necessary to make the application run. It uses the structure defined in the DeviceTypeDefinition and contains the behavior of the objects. Currently the CPC package provides the Siemens and Schneider Baselines and also the UnCPC package which represents the WinCC OA behavior of the CPC objects.
- Semantic Check rules: these are special kinds of

templates used to check the specification file.

- Input/Output Commissioning template: it's a special kind of template designed to generate a IOCommissioning file to help the process and control engineer during the commissioning of the system.

UNICOS MANAGEMENT

The special architecture of UAB, where the software is split in several packages with different life cycles requires an appropriate mechanism to manage the different software versions (for example, while the UAB Core will be very stable, the resource package is susceptible to be modified frequently, changes such as adding new devices, modifying existing devices or modifying the syntax of the output files). For this purpose, the UAB Bootstrap has been created. This tool, developed in Java, is used to manage the different UAB components and its resources packages. The main functionalities of this tool are:

- Upon first installation of UAB, it offers to download the last version of the available UAB components.
- Check for updates of the installed components and its compatible resource packages.
- Download/install new UAB components.
- Unique entry point to execute the UAB components installed.

The different pieces of software that compose UAB (core, plug-ins, resources) are packaged in different artifacts and deployed in a repository manager (Nexus). When the Bootstrap is executed, it queries the repository manager to discover new components, new versions of the installed components and new versions of the resource packages. When a new version is available on the server, it will notify the user and offer to download the new software.

To achieve this, the Bootstrap uses the *Nexus REST API* and the *Aether library*. The repository manager provides the REST API that can be used externally to query Nexus for available artifacts or test if any of the available artifacts has a version number higher than the installed ones. Aether is a general purpose library for interacting with artifact repositories. It provides functionalities to specify the repository locations, dependency resolution between artifacts and artifacts download.

Figure 6 shows the different steps to create an application for a CPC user. The UAB Bootstrap packs the selected CPC component version and the compatible CPC resources package from the Nexus server and installs the UAB software in a local machine. Thus the user will create a CPC application using the CPC wizard installed.

CONCLUSION

The UAB CPC6 component has been developed to add several improvements for developers and users (control system engineers).

The developer, using UAB, will obtain more flexibility and performance, allowing to add new CPC objects easily,

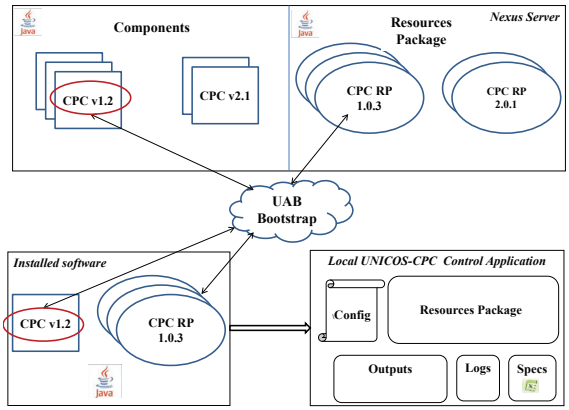


Figure 6: UAB Management.

new plug-ins for new platforms (for example the CIET plug-ins), templates or semantic check rules.

For the user, this new version offers a user-friendly graphic interface to develop CPC applications and a set of services like a powerful version management which guaranties the maintenance of these applications, logging, semantic check verifications, etc. The user can also add new objects, templates or check rules to the resources package.

REFERENCES

[1] Ph. Gayet, R. Barillère. "UNICOS a framework to build industry like control systems: Principles & Methodology". *10th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Genève (Switzerland), 10-14 Oct 2005.

[2] M. Dutour. "Software Factory Techniques applied to process control at CERN". *11th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Tennessee (USA), 15-19 Oct 2007.

[3] Hervé Milcent, Enrique Blanco, Frédric Bernard, Philippe Gayet. "UNICOS: AN OPEN FRAMEWORK". *12th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Grenoble (France), 12-16 Oct 2009.

[4] D. Willeman, E. Blanco, B. Bradu, J. Ortola. "UNICOS CPC new domains of application: Vacuum and Cooling & Ventilation". *13th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Grenoble (France), 10-14 Oct 2011.

[5] E. Blanco, A. Merezhin, B. Bradu, B. Fernández Adiego, D. Willeman, J. Rochez, J.M. Beckers, J. Ortola Vidal, Ph. Durand, S. Izquierdo Rosas. "UNICOS Evolution: CPC Version 6". *13th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Grenoble (France), 10-14 Oct 2011.

[6] B. Copy, R. Barillere, R.N. Fernandes, B. Fernandez Adiego, I. Prieto Barreiro. "Model Oriented Application Generation for Industrial Control Systems". *13th ICALEPCS Int. Conf. on Accelerator and Large Expt. Physics Control Systemns.* Grenoble (France), 10-14 Oct 2011.

EXPERIENCE IN USING BASED EMBEDDED CONTROLLERS WITH EPICS ENVIRONMENT FOR BEAM TRANSPORT IN SPES OFF-LINE TARGET PROTOTYPE

M.Montis, M.Giacchini, LNL INFN, Legnaro, Italy

Abstract

EPICS[1] was chosen as general framework to develop the control system of SPES facility under construction at LNL[2]. We report some experience in using some commercial devices based on Debian Linux to control the electrostatic deflectors installed on the beam line at the output of target chamber. We discuss this solution and compare it to other IOC implementations in use in the Target control system.

INTRODUCTION

SPES accelerator, under construction at the Legnaro National Laboratories (LNL), is on ISOL facility (Isotope Separation On-Line) dedicated to the production of radiactive ion beams with high energy and high degree of purity. In a structure like this one, the core of the project are the production target and the ionization ion system, which were carried out a study and design from scratch. This structure is currently in production for execution of experiments from the beginning of 2010.

Analyzing the characteristics required for inspection of this apparatus, it is possible to observe that:

- an heterogeneity of hardware and software solutions are required;
- we have to manage a distribuite control system;
- it is mandatory a real-time processing for every control variable, in order to ensure a robust system.

According to specific requests for software control, optimization of costs that derives from using an opensource software and favouring cooperation between laboratories, EPICS was chosen as control system software for the SPES facility.

The control network architecture created to manage the off-line target prototype is a multi-level structure composed by machines dedicated to network management and all EPICS hosts that implement the real control system environment. For managing all the devices used for the ion beam production, we decided to use PLC solutions in every sub-system in which safety is mandatory and it is necessary to implement fault-tolerant solutions, while particular embedded systems equipped with EPICS software manage all the devices dedicated to the ion beam manipulation (ion beam acceleration and ion beam focus)[3]. Appropriate solutions have been implemented for the integration of PLC subsystem within the EPICS environment, which it will explain in the next section.

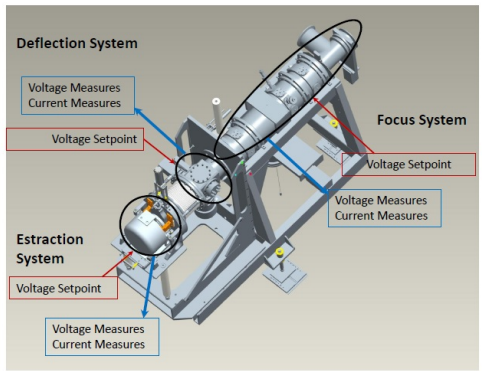


Figure 1: SPES Off-Line Target Prototype: systems and signals managed.

THE CONTROL SYSTEM

System Description

Managing the ion beam geometry, linearity and acceleration is made through the adjustment of the intensity of electrostatic fields generated by appropriate medium voltage power supplies (0-4kV) and high voltage power supplies (0-65kV). Controlling these power supplies it is possible to supervise the ion transport efficiency and optimize the emittance index. For this reason it is necessary to adopt appropriate control strategies to ensure a sufficient level of precision in the voltage settings, and security to all the devices managed.

We decided to base control algorithms on the following strategies:

- *voltage control*: initially we created a closed-loop system with PID regulators for controlling all the power supplies used in the deflection, focus and estraction systems. After this we changed the control strategy and we implemented a new control based in bisection algorithm because during the runs users had to usually change control parameters;
- *current monitoring*: the presence of any current could lead to the establishment of electrical discharges, resulting in the breakdown of components. For this reason we chose to implement a current monitoring strategy, with system shutting down when values exceeded a preset alarm threshold. Because of current values depends on power supplies work condition, we decided to define two different kind of thresholds:

  1. one current threshold for voltage's transitory;

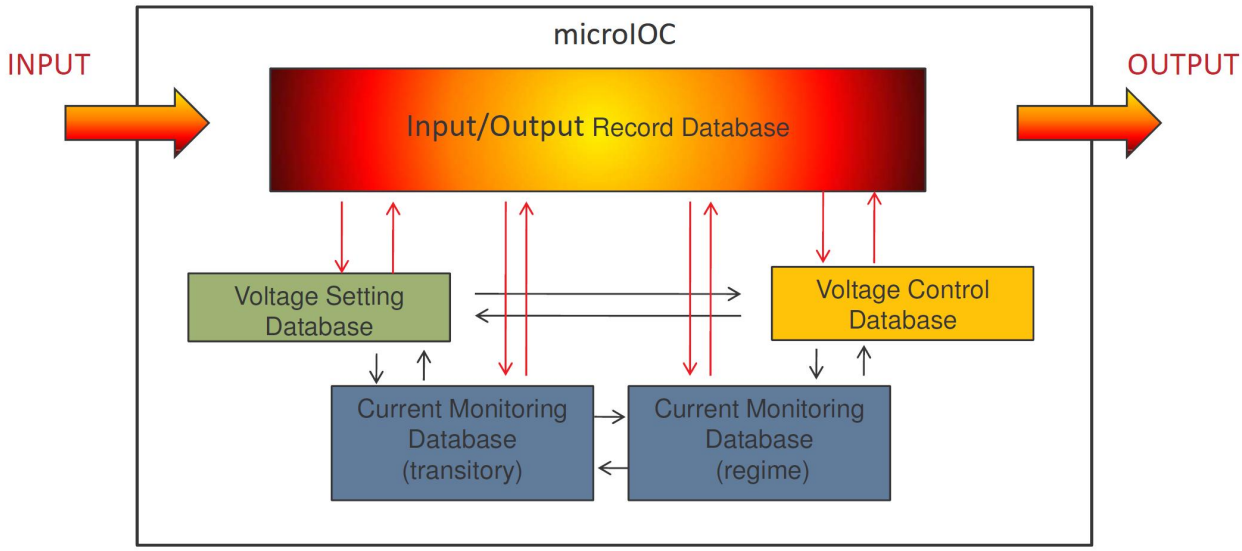


Figure 2: EPICS application developed: it is possible to see the connections between all the Record Istance Files defined and the signals form the field.

2. one current threshold when power supplies work in regime (costant voltage).

Appropriate EPICS Databases these strategy, and all the databases created for the power supplies remote control define the EPICS control system application used to manage the ion beam accelerator and focus systems for the SPES off-line target prototype.

Hardware Solution

While for the others sub-systems that constitute the SPES off-line target prototype we used computer general-purpose to control them, for the accelerator and optic sub-systems we decided to use particular embedded devices equipped with data acquisition cards and the software needed to define an EPICS Input/Output Controller (IOC).

The hardware chosen is an embedded computer (*microIOC*) based on the standard PC/104 and distributed by Cosylab, which allows for a robust and flexible remote control of devices connected to it, thanks to the absence of moving parts (such as hard drives and cooling fans) and a variety of communication interfaces available.

The Channel Access servers, used within the network for systems management of optical beam acceleration, are microIOC with the following characteristics:

- VGA connection;
- two ethernet connections, one in DHCP and one with static IP;
- two USB connestions;
- one RS232 connection;
- one AD I/O board (ADIO 104-AIO12-8) used for data acquisition and signal processing. It is possible to manage:
  - 16 digital input;
  - 16 digital output;
  - 24 analog input (12bit resolution);
  - 12 analog output (12bit resolution).

  Using this kind of boards we can control all the signals desired making an error of $\pm 1V$ for the medium voltage power supplies and $\pm 15V$ for the high voltage power supplies. While in the first case the error is acceptable, for the high voltage power supplies it is mandatory to decrease this value. One possibility suggested is to replace the ADIO board with another board having at least 16bit resolution for the analog input/output.

The operating system (a Debian based GNU/Linux custom distribution) and the EPICS software are stored in a Flash Card memory card. The EPICS environment is configured to initialize an IOC application at boot time. In this way the microIOC, when connected to the network, automatically creates the Channel Access Servers without further configuration by the user. The ADIO board uses a special *EPICS Driver Support* to interact with the control system environment and manage all the digital and analog signals.

During studies made and through the EPICS Community, we analized that the most used way to interface PLC systems with the EPICS environment is using particular hardware called OPC-Servers, that define a layer where the PLC signals are mapped in EPICS Process Variables. To optimize costs and minimize hardware complexity, we decided to implement it with a differet communication interface using the ADIO cards provided by microIOC.

For monitoring the embedded systems and the EPICS environment, we used an appropriate machine equipped with Nagios[4] software; in this way it is possible to control both the hardware and the software status.

Software Solution and Application Developed

During code development we tried to find a database structure which would allow to minimaze the number of Process Variables we have to manage and simplify the software maintenance. In this way it is possible to have a synthetic and readable code for anyone who have to monitor the SPES control system infrastructure. Following the project directives, the control strategy adopted and studies made with other EPICS applications we decided to develop a program structured in five different *Record Instance file*. Analizing Figure 2 shows:

- one Database, using the EPICS Driver Support, defines the interface between the EPICS envoronment and the AD I/O board mounted on the microIOC. This Record Istance File allows to manage all the inputs coming from the field and the outputs used to control the devices desired, such as indicated in Figure 3;
- one Database manages the Records used into the Graphical User Interface for setting all the power supplies's control parameters;
- one Database monitors the currents provided by the power supplies when they work on regime;
- one Database monitors power supplies's currents during voltage transitories;
- one Database realizes the voltage control strategy (PID regulator and bisection algorithm).

Realizing a modular structure for the software allows us to modify it in funcion of the experiment's requests (concerning the control strategies) and the technical requests (for example hardware maintenance) without changing the entire application: for example we modified the control strategy switching between two different databases (as rappresented in Figure 4) keeping unaltered all the other record istance files, except for fews links used to forward the process chain.

From an informatic point of view, Databases are built up with couples of template/substituion files: throught this EPICS environment feature, it is possible to modify all the Process Variables desired (number, type, attributes) and the

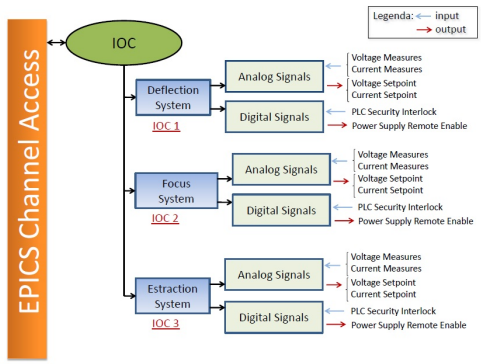


Figure 3: EPICS Control System Structure.

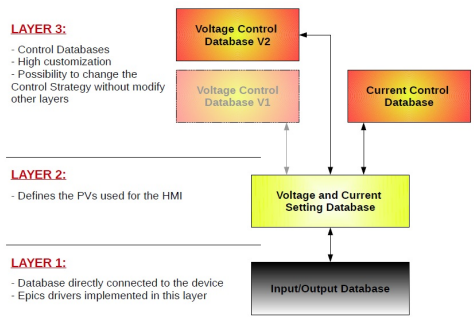


Figure 4: EPICS Application Database - Layers.

database architecture in a fast way editing a little set of text files; at the same time, if someone prefers use graphical editors (such as *VisualDCT*) to edit template files, the database structure will result easier to read and modify, because all the Records needed by the appllication will be instanced only at the IOC boot time. In this way both the debug phase and the maintenance are simplified; in particular this second feature is really important in a project like SPES, where all the works are developed and managed by different people.

For defining Process Variable names, we created a special Naming Convention that provides a descriptive name for every Record and, through it, every user can easy find and modify all the control parameters desired. This Naming Convention will be adopted for the entire facility's control system.

For every EPICS application developed, we created a specific Graphical User Interface (GUI) with Medm, an EPICS tool that provides an complete editor for creating and managing GUIs; an example of Medm GUI is visible in Figure 5. These interfaces are only used by software developers for managing the EPICS control system software, while specific GUIs developed in CSS are used by general users to supervise the SPES off-line target prototype and, in the future, all the SPES facility.

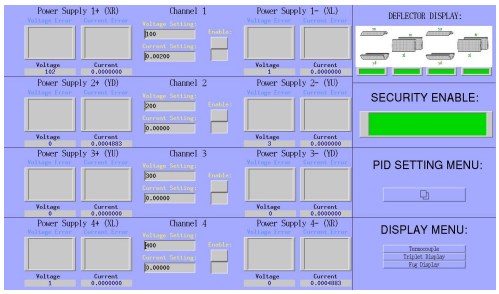


Figure 5: EPICS GUI developed for system maintenance.

CONCLUSION

Particular hardware and software solutions are adopted to design and develop the ion beam accelerator and focus systems in SPES off-line target prototype. On the hardware side, we decided to use specific embedded devices based on the PC/104 standard equipped with all the EPICS tools needed to connect them into the off-line target prototype's control system. During this production period we observed that this hardware solution guatantees good performances. This hardaware allows to create a trasparent layer between PLC systems and EPICS software too, minimizing possible incompatibilities between these two environments.

On the software side, the EPICS application structure adopted permits an easy management by users and a easy reuse of most of the code wrote for others applications and experiments, allowing a minimization of setting times. In general a modular approach for developping EPICS applications is desirable, especially in large and distribuited environment like the SPES facility.

REFERENCES

[1] EPICS – Experimental Physics and Industrial Control System `www.aps.anl.gov/epics/`.

[2] EPICS at Laboratori Nazionali di Legnaro `http://www.lnl.infn.it/~epics`

[3] M. Montis, Master's degree thesis, Università degli Studi di Padova, a.a. 2009-2010 `http://www.lnl.infn.it/~epics/THESIS/TesiMaurizioMontis.pdf`.

[4] NAL (Nagios Alarm Handler)(Not Alarm Handler) `http://www.lnl.infn.it/~epics/NAL.html`

SPIRAL2 CONTROL COMMAND: A STANDARDIZED INTERFACE BETWEEN HIGH LEVEL APPLICATIONS AND EPICS IOCS

C. Haquin, P. Gillette, E. Lemaître, L. Philippe, D. Touchard, Ganil, Caen, France
F. Gougnaud, Y. Lussignol, CEA/DSM/IRFU, Saclay, France.

Abstract

The SPIRAL2 linear accelerator [1] will produce entirely new particle beams enabling exploration of the boundaries of matter. Coupled with the existing GANIL machine this new facility will produce light and heavy exotic nuclei at extremely high intensities. The field deployment of the Control System relies on Linux PCs and servers, VME VxWorks crates and Siemens PLCs; equipment will be addressed either directly or using a Modbus/TCP field bus network. Several laboratories are involved in the software development of the control system. In order to improve efficiency of the collaboration, a special care is taken to the software organization. During the development phase, in a context of tough budget and time constraint, this really makes sense, but also for the exploitation of the new machine, it helps us to design a control system that will require as little effort as possible for maintenance and evolution. The major concepts of this organization are the choice of EPICS, the definition of an EPICS directory tree specific to SPIRAL2, called "topSP2", this is our reference work area for development, integration and exploitation, and the use of version control system (SVN) to store and share our developments independently of the multi-site dimension of the project. The next concept is the definition of a "standardized interface" between high level applications programmed in Java and EPICS databases running in IOCs. This paper relates the rational and objectives of this interface and also its development cycle from specification using UML diagrams to test on the actual equipment.

INTRODUCTION

The Spiral2 Control System is designed with a typical EPICS architecture, relying on OPI Clients and IOC servers communicating using Channel Access (CA) protocol.

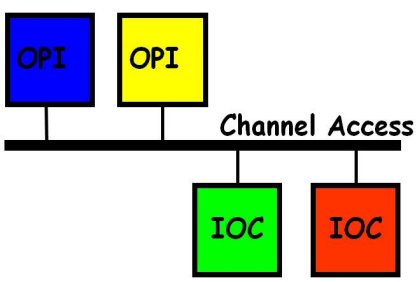


Figure 1: Spiral2 Control System Architecture.

The CA protocol allows OPI to read, write and monitor variables called Process Variables (PV) located in IOC. The CA protocol enable any OPI to access any PV as soon as it knows the PV name, there's no need to know which IOC hosts the PV.

OPI issues CA requests to IOC, which eventually interact with equipment to perform the actual read or write operation. This is the mean by which, over CA, functions are provided to OPI to fulfil its control tasks.

Hence, OPI need to know the names of the PV that correspond to its purposes. This can quickly become a big mess on OPI side since there are many type equipment to be driven and each developer could to adopt its own philosophy on both OPI and IOC sides.

It then appeared obvious that design should be optimized in order to reduce the development effort, but also in the machine exploitation perspective, to be able to face the evolution and maintenance requirements with a small team. Consequently, the decision was made to homogenize the way OPI control the various equipments through PV.

So, starting from the fact that PV names are almost completely determined by the naming convention, which take into account the localisation and the type of equipment [2], and the observation that equipment driving is always achieved through same kind of functions, we started to glimpse the standard interface concept in the sense that the naming convention should be pushed one step further in order to codify the remaining part of the PV name in correlation with the expected function.

This paper explains how the Standard Interface specifies the naming of the PV through which functions are provided to OPI, how it has been implemented on EPICS Data Base side (DB) and the first feedback is presented and next steps are envisaged.

SPECIFICATION OF THE STANDARD INTERFACE

Presentation

- Takes place in CA communication between OPI and IOC.
- Takes advantage of the fact that all equipment are drivable by the same kind of functions.
- Relies on a set of interface PV through which IOC provide the functions needed by OPI.

Goals

- Reduce development effort
- Reduce maintenance effort
- Ease component reuse

Functionalities

We noticed that on OPI side, independently from the equipment type, the following functions are always required:

- Write Consigns
- Write Commands
- Read Back Consign
- Read Measurements
- Read States
- Read Defects

PV Rules

Since functionalities are provided through PV, the name is related to the expected function. The rules are such that OPI can simply deduce the name of the needed PV from the needed functionality.

In the following, "$(CODIF)" refer to the first part of the PV name determined by the project naming convention, "PType" refer to the type of parameter to control (example I for current, V voltage).

Rules for consign related PV

Consign handling consist in write, read back, read actual value operations, each operation being handled by a record.

Table 1: PV Naming Rules for Consign Handling

Operation	PV Name
Write	$(CODIF):*PType*Cons
ReadBack	$(CODIF):*PType*ConsLect
ReadActual	$(CODIF):*PType*Act

In addition, filling of operator display fields is mandatory. In case of *PType*Act PV, alarm fields related to limit and severity are also mandatory.

Rules for measurements related PV

Measurement handling consists only in read actual value operation.

Table 2: PV Naming Rule for Measurement Handling

Operation	PV Name
ReadActual	$(CODIF):*PType*Meas

In addition, filling of operator display fields and alarm fields related to limit and severity is mandatory.

Rules for Commands PV

Commands are orders issued used for example to reset an equipment or to put it "on/off", "in/out" … It basically consists in a write operation, but it is also convenient for OPI to retrieve the list of possible commands, this is achieved with a mbbo record in which possible commands correspond to its ENUM strings. The whole command list can then be retrieved with a caget –d 31 *PVName* and a command can be issued by writing one of the string in the same PV.

Table 3: PV Naming Rule for Command Handling

Operation	PV Name
ReadActual	$(CODIF):Cmd

Rules for status related PV

Status handling consists in read operations related to the states and defects of the system. State and defect words must be analyzed bit per bit, each bit being associated with a string giving its signification. To avoid repetition of the same analysis on OPI side, this analysis is performed on IOC side. The result is provided through the standard interface with a record developed for this purpose. The record is in charge of words analysis and hosts all standard interface compliant PV.

Table 4: PV Hosted by Status Record

Operation	PV Name
Read	$(CODIF):Status.STATE_WORD
Read	$(CODIF):Status.STATE_ON_LIST
Read	$(CODIF):Status.STATE_OFF_LIST
Read	$(CODIF):Status.STATE
Read	$(CODIF):Status.DEFECT_WORD
Read	$(CODIF):Status.DEFECT_LIST
Read	$(CODIF):Status.DEFECT

In the previous table, the xxx_WORD PV contain a word to analyse, xxx_LIST contain a list of strings describing each bit of a word. The STATE PV contains the result of the analysis of the STATE word, when an OFF bit ('0') is found, its corresponding OFF string is copied from the STATE_OFF_LIST, when an ON bit ('1') is found, the string is copied from the STATE_OFF_LIST. The DEFECT PV contains the result of the analysis of the DEFECT word, only ON bits are treated, when an ON bit ('1') is found, its corresponding ON string is copied from the DEFECT_LIST. Each xxx_LIST PV can be retrieved with a caget –d 31 *PVName*

IMPLEMENTATION

This section explains how the standard interface is integrated in EPICS DB.

Consign and Measurements

A read interface PV is handled by an ai or longin record and a written interface PV is handled by an ao or longout record.

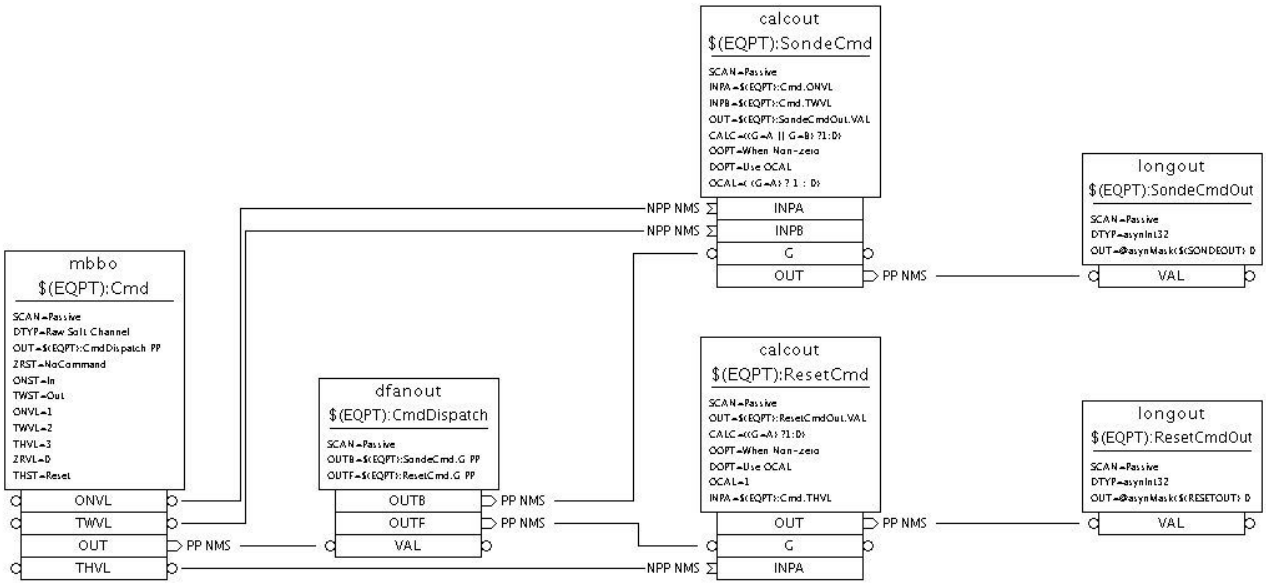


Figure 2: Example of commands handling implementation for an equipment having three commands: "in/out", to be written in the same register and "reset" being written in its own register. When a command is received by the mbbo record, it is transmitted to a dfanout record which dispatches it to two calcout records. The SondeCmd calcout record checks that the command is "in" or "out" if yes the command is actually written via the corresponding longout record, The ResetCmd calcout record checks that the command is "reset" if yes the command is actually written via the corresponding longout record.

Commands

The command interface PV is hosted by a mbbo record, the strings for each individual command are filled in the ZRST, ONST … fields. However, the mbbo record doesn't perform the write operation, this is done by longout record, there are one longout record per command register, depending on the driven equipment. In addition, the path between mbbo record and longout record(s) is secured by calcout record(s) in charge of actually activating or not the longout record(s) and ensuring the validity of what is actually written.

States and Defects

The Status record is in charge of internal state and defect words analysis and interface PV update. The Status record is basically a genSub record in which output fields were renamed and configured in order to comply with the standard interface. The challenging part was to manage each equipment type with its own set of strings through the common to all interface PV. The retained solution consists in defining a version of the Status record specific to each type of equipment, this is simply achieved by creating a copy of the record support structure renamed according to the new equipment and by providing a ".dbd" file in which the equipment strings are defined, in this manner the link between an equipment type and its set of strings is automatically established. Though there are several version of the Status record, they all share the same source code.

At processing time, the record fetches internal state and defect words by dedicated inputs and trigs the processing for each word, if the value has changed. The word analysis process is the same for both state and defect words, a parameter specifying the expected behaviour: "ON" and "OFF" bit values analysis or just "ON" bit value analysis. Strings specified in the ".dbd", basically available in VDCT menus are retrieved with the staticLib functions, and made available to the word analysis process to build the STATUS and DEFECT strings array.

CONCLUSION

We just ended the definition and development of the Standard Interface, so its deployments is still limited to power supply, profiler and faraday cups and the Status Record is not integrated yet. But we used it when setting up and running the control system for deutons source test at Saclay, in a "debug" context it turns out to be really efficient since with the naming convention and the command through mbbo record, driving equipment with simple CA directives in a terminal is a child's play, we can foresee it will ease OPI development. The next step is to integrate the Status record in the power supply, profiler and RF equipments [3]. And soon as possible, all EPICS modules will be made Standard Interface compliant.

REFERENCES

[1] http://www.ganil-spiral2.eu/spiral2-us/keys-sp2/whats-spiral2
[2] D. Touchard "Prel The Spiral2 Control software organization and management" Icalepcs 2009.
[3] D. Touchard "The Spiral2 Radio Frequency Command Control" Icalepcs 2011.

COMMERCIAL FPGA BASED MULTIPURPOSE CONTROLLER: IMPLEMENTATION PERSPECTIVE*

I. Arredondo, M. del Campo, P. Echevarria, D. Belver,
L. Muguira, N. Garmendia, H. Hassanzadegan, M. Eguiraun, ESS-Bilbao, Spain
J. Jugo, V. Etxebarria, University of the Basque Country, Leioa, Spain

Abstract

This work presents a fast acquisition multipurpose controller, focussing on its EPICS integration and on its XML based configuration. This controller is based on a Lyrtech VHS-ADC board which encloses an FPGA, connected to a Host PC. This Host acts as local controller and implements an IOC integrating the device in an EPICS network. These tasks have been performed using Java as the main tool to program the PC to make the device fit the desired application. All the process includes the use of different technologies: JNA to handle C functions i.e. FPGA API, JavaIOC to integrate EPICS and XML w3c DOM classes to easily configure the particular application. In order to manage the functions, Java specific tools have been developed: Methods to manage the FPGA (read/write registers, acquire data, ...), methods to create and use the EPICS server (put, get, monitor, ...), mathematical methods to process the data (numeric format conversions, ...) and methods to create/initialize the application structure by means of an XML file (parse elements, build the DOM and the specific application structure). This XML file has some common nodes and tags for all the applications: FPGA registers specifications definition and EPICS variables. This means that the user only has to include a node for the specific application and use the mentioned tools. A main class is in charge of managing the FPGA and EPICS server according to this XML file. This multipurpose controller has been successfully used to implement a BPM and an LLRF application for the ESS-Bilbao facility.

INTRODUCTION

The utilization of the same hardware for the implementation of several applications has some advantages such as, the knowledge of how it works, its limitations, an easier maintenance and development of new tools.

Normally each piece of hardware has its own purpose but with the popularization of FPGAs the hardware has become reconfigurable. The development of an FPGA based hardware could take a long time and, therefore, the utilization of commercial hardware is beneficial. This choice contributes with a fast implementation and reliability, because commercial hardware is, usually, well tested.

In this work the VHS-ADC board of Lyrtech company [1], combined with a PC are used to design a multipurpose controller focusing their function on particle accelerators applications.

* iarredondo@essbilbao.org

In particle accelerators the most of the devices have to be connected to a central network to be controlled or monitored via the facility's standard middleware. Concretely in this paper EPICS [2] is implemented.

On the other hand, and following with the idea of the fast implementation and easy maintenance, it is important to provide some tools to make easier to the programmer the implementation of each application and to allow the user configuring the hardware without the necessity of programming. Therefore, some libraries have been designed to help the programmer and XML language has been chosen to interact with the user.

There are two main processes which are performed in the applications: Monitor and Control. Monitoring process is summarized in read data from the experiment with the hardware, access to this information and publish it on the network. On the other hand, the process of control can be described as a monitoring process with the possibility of interacting with the experiment. This is performed by changing a parameter, sending it through the network to a hardware controller and write it in the hardware.

In the following sections it will be described how to do the implementation of these two processes in a reconfigurable way with a minimal programming by the final user.

SYSTEM OVERVIEW

The main idea of the system is to use a commercial FPGA, the VHS-ADC board of Lyrtech company, to manage different applications related to control and monitor processes and data logging in large facilities. To drive the FPGA, a Host PC is utilized. This PC implements the Hardware Controller (HC) which is the link between the FPGA and the user.

The structure of the system is depicted in Figure 1. The instrument under control is connected to the Lyrtech board which is linked to the Host PC via a cPCI port. The HC reads and writes parameters from and to the FPGA's registers and publishes them over the network by means of an EPICS server. Then, EPICS clients can use the data and change the control parameters.

The structure of the conceptual programming is the following: Acquisition, fast calculations and real-time process are handled by the FPGA while slow or heavy calculations, data logging and EPICS related functions are performed in the HC.

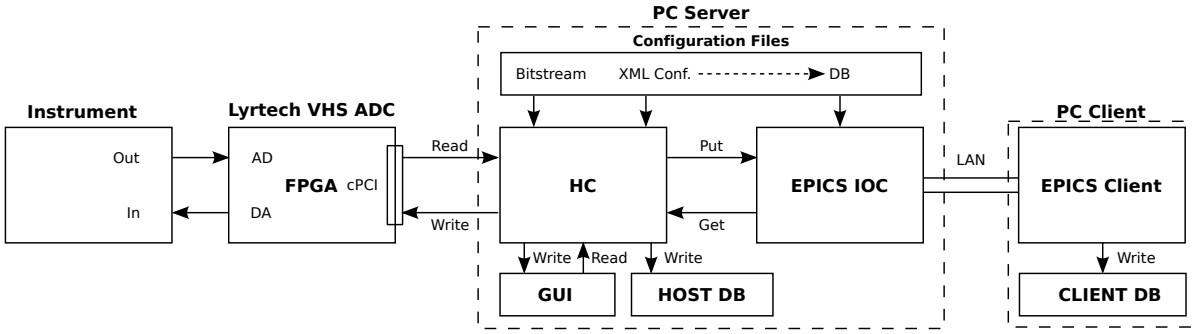


Figure 1: General diagram.

HARDWARE CONTROLLER IMPLEMENTATION

The HC is the core of the system since it is in charge of managing the FPGA, the EPICS server, the EPICS IOC, the DB, the XML configuration files, the local user GUI and of performing some calculations. The integration of all these devices has been carried out utilizing Java language because it is able to handle all of the mentioned needs. Each case is detailed as following:

FPGA: It uses a C programmed API, so JNA [3] is utilized.

EPICS Server and IOC: There is a Java based tool called JavaIOC [4].

DB: It is mainly handled by the java.sql class, because it is implemented in a MySQL database [5].

XML: The configuration file is relatively small and a fast access to the data is required. DOM technology from w3c [6] has been chosen.

GUI: SWT toolkit [7] is used.

The HC program is built with several threads in parallel, to be able to read and write from/to the FPGA, create the EPICS server, update the DB, build the GUI and parse the XML configuration file[1].

However, in order to guarantee versatility and the ease of programming different applications to the final user, some tools, xml rules and instructions have to be provided. These are treated on the following sections.

Managing Tools

In order to provide a set of tools to program the part of the HC which is related with the specific application, some Java classes which fit most common necessities have been designed. These are:

org.essb.mc.epics.db: EPICS Archiver MySQL DB implementation, managing and configuration tools.

org.essb.mc.epics.db.tables: EPICS Archiver MySQL DB formatting utils.

org.essb.mc.epics.utils: EPICS utils to manage a javaIOC: create/destroy context, connect/disconnect channels, caget synchronous/asynchronous, caput synchronous/asynchronous, create/destroy monitor and camonitor.

org.essb.mc.fpga.program: Handle the FPGA: Open/-Close board, program FPGA, program Flash, set FPGA clock, set ADCs status, read ADCs overflow and Read-/Write Registers.

org.essb.mc.fpga.maths: FPGA raw to engineering units conversion and vice versa, and standard numeric conversions.

org.essb.mc.gui.general: Utils to create the GUIs with the most common objects.

org.essb.mc.xml: Tools to acquire the configuration data from an XML document and use it in the main program.

These tools are enough to program a very wide range of applications. Nevertheless once one is set up and working, it is better to have an easy way to reconfigure some parameters depending on the specific device or situation. Here is when the XML configuration file makes the MC more versatile.

XML Based Configuration

With the XML configuration file, the goal is to reconfigure the HC to manage the desired application. Therefore, the first action taken by the HC when it is started is to ask the user for a configuration file. Then it is parsed and dumped in a well defined object in the program. All the necessary tools to perform this issue are in the org.essb.mc.xml class.

The time constraints are, normally, a common characteristic in the accelerators applications, hence, it is important to guarantee a fast access to the data of the configuration file. For this reason, it is better to have this data in the RAM and, therefore, DOM technology is very suitable.

A typical configuration file is like the following one:

```
<!DOCTYPE MCEBC [ <!-- Multipurpose
    Controller Ess Bilbo Configuration-->
  <!ELEMENT MCESSB (Name, Value, FPGA, EPICS,
    BPM)><!-- Multipurpose Controller ESS
    Bilbo-->
  <!ELEMENT Name (#PCDATA)>
```

[1] If more information about the structure of the HC is need, it could be found in [8].

```
<!ELEMENT Value    (#PCDATA)> <!--
    Initial Value -->
<!ELEMENT FPGA   (Used,Bits,Prec,Reg,
    Custom,Custom_ADDR,Signed,RW)>
  <!ELEMENT Used     (#PCDATA)>  <!-- Is
      used in FPGA ?-->
  <!ELEMENT Bits     (#PCDATA)>
  <!ELEMENT Prec     (#PCDATA)>
  <!ELEMENT Reg      (#PCDATA)>
  <!ELEMENT Custom   (#PCDATA)> <!--
      Custom => true or Lyrtech Reg =>
      false -->
  <!ELEMENT Custom_ADDR    (#PCDATA)> <!
      -- Custom register Address -->
  <!ELEMENT Signed   (#PCDATA)> <!--
      Signed => true or No signed =>
      false -->
  <!ELEMENT RW   (#PCDATA)> <!-- Read/
      Write=> 0,1 -->
<!ELEMENT EPICS (Used)>  <!-- EPICS
    config if vble is used in EPICS -->
  <!ELEMENT Used   (#PCDATA)>  <!-- Is
      used in EPICS ?-->
<!ELEMENT BPM (Used,Element,Fit)>  <!--
    BPM config -->
  <!ELEMENT Used   (#PCDATA)>  <!-- Is
      used in BPM ?-->
  <!ELEMENT Element (#PCDATA)> <!--
      Indicate which is the variable in
       BPM application. Left, Right, Up
      , Down (buttons), Amp, Phase, X
      or Y -->
  <!ELEMENT Fit  (#PCDATA)> <!-- BPM
      nonlinear fitting values config
      -->
]>
```

There are five well defined fields into the XML file. The first four ones are compulsory and are repeated for all applications. These are: The name of the variable, its initial value, the FPGA's register configuration and if it is EPICS published or not.

On the other hand, there is the application field which has to be detailed by the user. In the example there is a BPM structure which has fields linking/explaining which button is measuring the variable and some values to fit the electrical delays.

The implemented data structure for the reconfigurability of the HC combines the DOM data calls, with a vectors structure which contains the index of the elements with the same characteristics. For example, in one vector it is stored the index of the elements of the DOM structure which have to be published in EPICS. The implemented structure is in Figure 2. The vectors (Value, FPGA/Read, FPGA/Write and EPICS) contain the index of the DOM node which fits with its function and, on the other hand, there are some structures which will contain the needed information associated with the vector. Conceptually, the working mode is

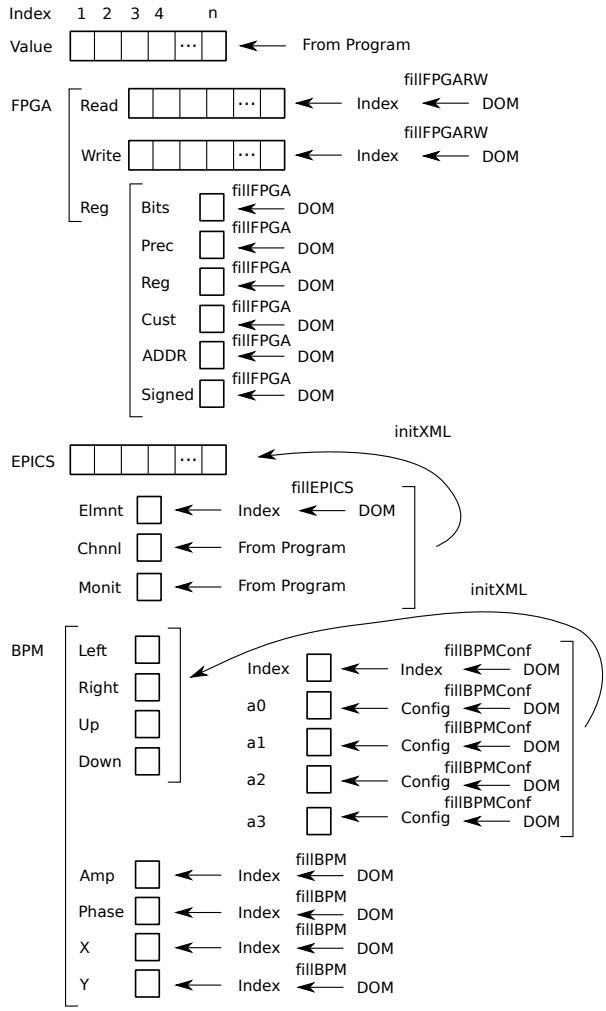


Figure 2: General diagram of communication between the control algorithm and the Host PC.

the following: Firstly the vectors are filled with the corresponding DOM node index. Then, when the program requires some data, it searches inside the vector, fills the corresponding structure with the tags of the node and uses this structure. In Figure 2 it is also detailed the method which is used to handle the data from the DOM.

Implementing a New Application

The steps to implement a new application assuming that the FPGA is suitable for the case (range of signals, ...) are the following:

1. Create the bitfile of the FPGA which fit with the application.
2. Update the configuration file:
   (a) Update the FPGA structures according to the previously designed FPGA bitfile.
   (b) Update the EPICS structures.

(c) Include a new structure for the application in the configuration file.

3. Use the org.essb.mc.xml tools to integrate the new structure into the HC.

4. Use the org.essb.mc.gui.general tools to adequate the GUI to the application. It is only needed to change the application tab, because the FPGA handling one is always the same.

CONCLUSIONS AND FUTURE WORK

In this work, a multipurpose controller based on a commercial FPGA has been presented. It has been designed to fit the large scientific facilities' requirements, using EPICS standard. Therefore, it has to be able to: Perform very fast calculations and precise acquisition, be integrated into the global control network and store the required data. The proposed solution, integrates an FPGA, an EPICS connection and a MySQL database in order to fulfill all of these specifications.

The key idea of this paper has been to explain how this controller has been made versatile, designing programming tools and using XML technology. Concretely, Java tools have been built to ease the labour of the programmer and an XML reconfiguration file has been integrated to make the final user able to configure the device.

Also it has been proposed a method to implement a new application based on the defined tools.

As future work it is foreseen the integration of EPICS database, which is an XML file, into the main XML configuration file. To perform this integration, the use of XLST is expected.

REFERENCES

[1] Lyrtech, http://www.lyrtech.com/.

[2] EPICS, http://www.aps.anl.gov/epics/.

[3] Java Native Access, https://jna.dev.java.net/.

[4] JavaIOC, http://epics-pvdata.sourceforge.net/.

[5] MySQL, http://www.mysql.com/.

[6] XML DOM, http://www.w3.org/DOM/.

[7] SWT, http://www.eclipse.org/swt/.

[8] I. Arredondo et al. "Fast acquisition multipurpose controller with epics integration and data logging", In *IPAC'11*, San Sebastian, Spain, 2011.

FUNCTION GENERATION AND REGULATION LIBRARIES AND THEIR APPLICATION TO THE CONTROL OF THE NEW MAIN POWER CONVERTER (POPS) AT THE CERN CPS

Q. King, S. T. Page, H. Thiesen, CERN, Geneva, Switzerland
M. Veenstra, EBG MedAustron, Wr. Neustadt, Austria

Abstract

Power converter control for the LHC is based on an embedded control computer called a Function Generator/Controller (FGC). Every converter includes an FGC with responsibility for the generation of the reference current as a function of time and the regulation of the circuit current, as well as control of the converter state [1]. With many new converter controls software classes in development it was decided to generalise several key components of the FGC software in the form of C libraries: function generation in libfg, regulation, limits and simulation in libreg and DCCT, ADC and DAC calibration in libcal. These libraries were first used in the software class dedicated to controlling the new 60MW main power converter (POPS) at the CERN Proton Synchrotron (CPS) where regulation of both magnetic field and circuit current is supported. This paper reports on the functionality provided by each library and in particular libfg and libreg. The libraries are already being used by software classes in development for the next generation FGC for Linac4 converters, as well as the CERN SPS converter controls (MUGEF) and MedAustron converter regulation board.

FUNCTION GENERATION LIBRARY

Calculating a reference value as a function of time is a key requirement for most regulated systems. Libfg is a C library that can generate ten different types of reference function, as listed in Table 1. For each type there are two C structures; one for the configuration of the reference function (config) and the other containing the parameters (pars) needed to generate the reference at a given time. Figure 1 shows how these structures relate to the associated C functions *Init()* and *Gen()*.

To summarise, the *Init* function processes the config structure and calculates the contents of the pars structure. This is then used by the *Gen* function to calculate the reference at the specified time.

Two other structures are concerned: limits and meta. Limits are provided to the *Init* function and allow the absolute, rate of change and acceleration limits to be checked. Meta is produced by the *Init* function with summary information (duration, min/max and start/end reference values). The *Init* function also needs to know the initial time delay and in most cases the initial reference value since most reference functions are relative to an initial reference. Only TABLE and SPLINE define absolutely the entire function and do not need it.

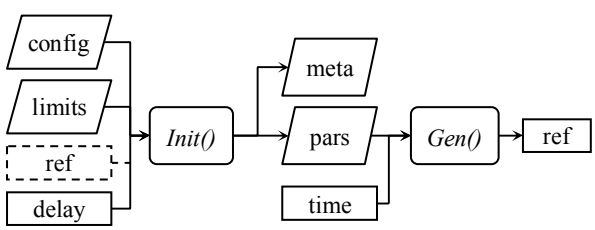


Figure 1: libfg data flow.

Table 1: Libfg Function Types

Name	Function type
PLEP	Single parabola-linear-exponential-parabola segment
PPPL	Multiple parabola-parabola-parabola-linear segments
TABLE	Table interpolated with linear segments
SPLINE	Table interpolated with parabolic splines
LINEAR	Single linear segment in a given time
CUBIC	Single cubic segment in a given time
SINE	Sine with optional window for smooth start/end
COSINE	Cosine with optional window for smooth start/end
SQUARE	Offset square wave
STEPS	Staircase of one or more steps

Reference and Time Units

Libfg does not dictate the interpretation of the reference. For magnet circuit control it might be voltage, current or field, depending upon the regulation mode. Similarly, libfg does not dictate the units of time, though at CERN the units are always seconds.

PLEP

The PLEP function provides a standard way to change the magnet circuit current. The initial and final parabolas provide a smooth start and end, which is vital for super-conducting circuits. The linear section takes over if the rate limit is reached and the exponential section may be needed when ramping down with a 1-quadrant converter. These are unipolar in current and voltage and so the current cannot be ramped down faster than the natural decay rate for the circuit ($T_c = L/R$). Thus when a PLEP is initialised, the result might be a P-P, a P-L-P, a P-E-P or a P-L-E-P depending upon the parameters and the difference between the initial and final reference values.

The PLEP function is special amongst all the functions in libfg because it can be initialised with a non-zero initial rate of change. This can be important if a running

function needs to be aborted smoothly, which is the case in the LHC if cryogenics report an impending problem.

PPPL

The PPPL reference was created for the CPS main magnet controls. The field is ramped up in stages with a series of linear plateaus defined parametrically using seven values. These specify a fast parabolic acceleration followed by a slow parabolic deceleration, then a fast parabolic deceleration and finally a linear section that is not necessarily constant.

TABLE

TABLE provides linear interpolation between points and is the main function type for LHC current references. Linear interpolation is sufficient provided points are close together when the reference is changing rapidly. If the reference should follow a parabola then equation (1) is a useful formula that gives the point spacing t as a function of the maximum acceptable interpolation error ε and the parabolic acceleration a:

$$t = 2\sqrt{(2\varepsilon/a)} \qquad (1)$$

SPLINE

SPLINE offers a way to use fewer points than TABLE by using parabolic splines rather than linear interpolation.

TRIM Functions

Trim functions are useful for small changes in the reference, especially when many circuits must change synchronously, since unlike the PLEP function, the duration for the change is an input parameter. CUBIC trims are essential for super-conducting circuits because they avoid discontinuities in the rate of change which would generate voltage spikes. LINEAR trims can be used for warm circuits but they suffer from undershoot and overshoot and are not used in the LHC.

TEST Functions

The SINE, COSINE, SQUARE and STEPS functions are very useful for converter testing and in some cases for accelerator measurements (e.g. K-modulation).

Definition of Time

Libfg uses single precision floats for time (and all other reference related parameters). The resolution of IEEE 32-bit floating point values is limited by the 23-bit mantissa and will allow 1 µs granularity for up to 10 s, 10 µs granularity up to 100 s and so on. If finer granularity is required then the library can be modified and recompiled to use double precision floats.

REGULATION LIBRARY

Libreg provides all the key components needed to write a program to regulate magnetic field or circuit current.

Magnet Load Model

Figure 2 shows the load model used by libreg. It is based on a magnet with inductance L and resistance R_m (zero for superconducting magnets). This is associated with a parallel damping resistor R_p and a series resistor R_s which corresponds to the cables or bus bars leading to the magnet. In many case multiple magnets are connected in series but for low frequencies their individual impedances can be combined into the four values given in figure 2.

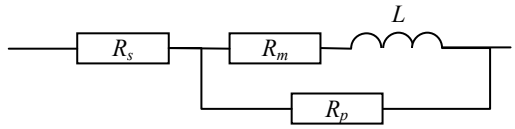


Figure 2: libreg load model.

The gain response of this model is first order and is shown in figure 3.

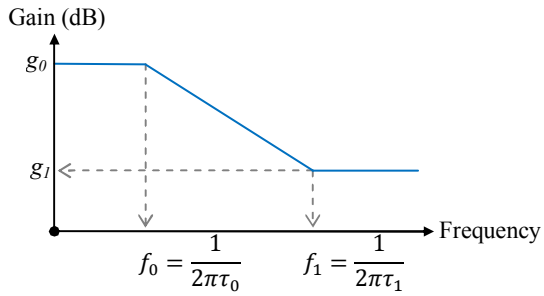


Figure 3: Load model Bode plot.

The pole and zero gains and time constants associated with the break frequencies in the Bode plot are given by:

$$g_0 = \frac{1}{R_s + \frac{R_p R_m}{R_p + R_m}} \qquad g_1 = \frac{1}{R_p + R_s} \qquad (2)$$

$$\tau_0 = \frac{L}{R_m + \frac{R_p R_s}{R_p + R_s}} \qquad \tau_1 = \frac{L}{R_p + R_m} \qquad (3)$$

The gains g_0 and g_1 correspond to the effective load resistance at zero and infinite frequency, while the time constants τ_0 and τ_1 correspond to the effective circuit resistance as seen by the inductance for shorted and open circuits.

Reference and Measurement Limits

Libreg provides functions that implement limits on the current or field and voltage references and the current or field measurement. These have different objectives. The measurement limit is used for protection of the circuit; the converter will be tripped if the limit is exceeded. The reference limits are associated with constraints in the converter and circuit and may cause the reference to be clipped. Normally limits are applied by libfg when initialising a reference function, however if a real-time reference is included then real-time limits are required.

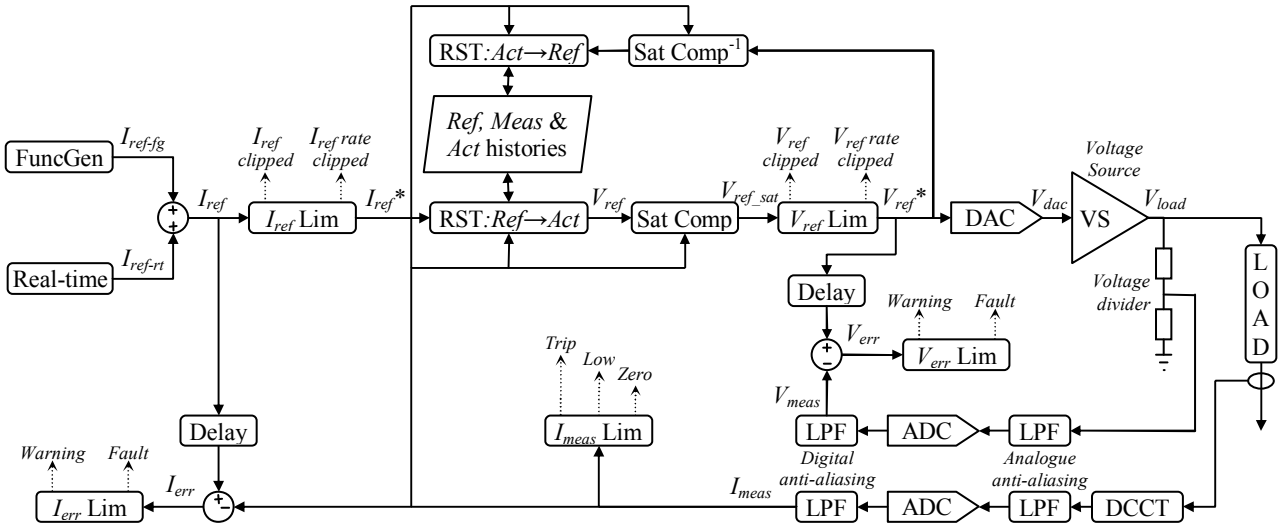


Figure 4: Overview of current regulation of an analogue voltage source.

RST Regulation

Figure 4 presents an overview of the function generation and regulation that can be implemented using libfg and libreg. One of the most flexible ways to implement a linear regulator is with the RST equation:

$$\sum_0^N\{Act_i \cdot S_i\} = \sum_0^N\{Ref_i \cdot T_i\} - \sum_0^N\{Meas_i \cdot R_i\} \qquad (4)$$

Where *i=0* corresponds to the current sample, *i=1* is the previous sample and so on. This notation was proposed by Landau [2], however in many text books the R and S polynomials are exchanged. By keeping the history of the previous *N* samples of the reference, measurement and actuation it is easy to calculate the new actuation if you know the new reference and measurement:

$$Act_0 = \frac{\sum_0^N\{Ref_i \cdot T_i\} - \sum_0^N\{Meas_i \cdot R_i\} - \sum_1^N\{Act_i \cdot S_i\}}{S_0} \qquad (5)$$

Equally, when the actuation is limited or is being driven in open-loop then it is equally easy to back-calculate the reference which, when combined with the new measurement will result in this actuation:

$$Ref_0 = \frac{\sum_0^N\{Act_i \cdot S_i\} + \sum_0^N\{Meas_i \cdot R_i\} - \sum_1^N\{Ref_i \cdot T_i\}}{T_0} \qquad (6)$$

In this way the reference history can be kept coherent with the measurement and actuation histories. This is equivalent to the anti-windup feature of a traditional regulation algorithm.

The benefit of the RST equation is that any linear regulator up to order *N* can be implemented by choosing the appropriate RST polynomial coefficients. Simple PI, PID, or PII controllers can be implemented [3] as well as more complex higher order systems, *without changing the software*. Of course, for complex higher-order systems calculating the coefficients is a challenge.

For a magnet circuit the reference and measurement can either be of the circuit current or magnetic field. The actuation defines the circuit voltage which the voltage source must try to follow.

If the required bandwidth of the current or field regulation is much less than the bandwidth of the voltage source and the bandwidth of the reference is lower still then a deadbeat controller can be used. Since this is exactly the case for the circuits in the LHC the regulation library includes a function to calculate the RST coefficients that will implement a deadbeat PII controller. This works extremely well in the LHC with negligible tracking error.

For applications that require as much bandwidth as possible the transfer function of the voltage source must be accurately modelled and a different approach taken to derive the RST coefficients. Matlab has a powerful toolkit for this, however it does require expert knowledge of automatic control theory.

Magnet Saturation Model

If warm magnets are used beyond about 0.9 teslas, their iron yokes will enter saturation and the inductance drops by as much as 60%. The corresponding change in time constant can destabilise the regulation loop if it is not compensated. Libreg supports compensation of this effect based on the simple linear model of the magnet inductance L_m(I) shown in figure 5. Three parameters are used to characterise the model: L_{sat}, I_{sat_start} and I_{sat_end}.

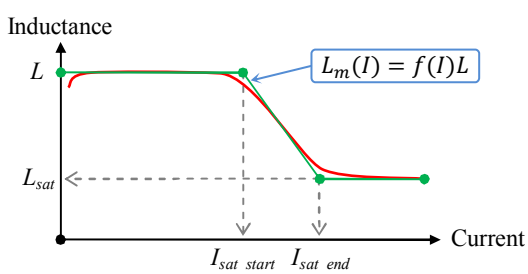


Figure 5: Magnet saturation model.

The saturation model is used to transform V_{ref} in order to hide the saturation effect from the RST algorithm:

$$V_{ref\_sat} = \{1 - f(I)\}IR + f(I)V_{ref} \quad (7)$$

Where $f(I) = L_m(I)/L$ and R is the load resistance associated with magnet load pole:

$$R = R_s + \frac{R_p R_m}{R_p + R_m} \quad (8)$$

As shown in figure 4, equation (7) is used to derive V_{ref_sat} from V_{ref} which is then limited to arrive at V_{ref}*. If clipping occurs then equation (7) is inverted to back-calculate V_{ref} from V_{ref}* and equation (6) is used to back-calculate the reference to keep the RST histories coherent.

Note that when regulating magnetic field, saturation is a second order effect and compensation is not required.

Regulation Error

Figure 4 shows how the regulation error is calculated from the difference between the delayed reference and the measurement. The delay must be set to the tracking delay of the regulator. Libreg includes functions that allow the error to be calculated and limits to be applied.

Simulation of Voltage Source and Load

It is highly desirable to build into converter controls software a simulation mode in which the voltage source, load, and measurements are simulated in software. Libreg includes functions that support a third-order model of the voltage source and simulation of the load model presented above.

FGTEST PROGRAM

A program called fgtest was written under Linux to test libfg and libreg. It is a template for a real-time converter control program, showing how the core elements of the regulation loop presented in figure 4 can be implemented. The program is driven from the command line using parameter files and writes signals to stdout as comma separated values that can be analysed with any suitable tool such as GNUplot, Matlab or Excel.

LIBCAL

Libcal contains functions to support the calibration of voltage dividers, DCCTs, ADCs and DACs, including the option for up to second order temperature compensation for ADCs and DCCTs. More information and the library source files can be downloaded from http://cern.ch/cclibs.

APPLICATIONS

Libfg and libreg were created to support multiple platforms and applications. So far they have been compiled on Linux PCs and TMS320C32 and TMS320C6727 DSPs and from 2012 they will be used in the converter controls software for the CERN LHC, SPS, CPS and PSB accelerators. They will also be used in the new converter controls for the MedAustron accelerator.

For now, only one converter in the CPS accelerator is controlled using the libraries; the new 60 MW POPS converter that drives the main magnets. This is a particularly interesting case since it can regulate magnetic field or circuit current on a cycle by cycle basis. The magnets saturate dramatically above 3.6 kA with a 60% reduction in inductance by the maximum current of 6 kA. This is successfully compensated when regulating current using equation (7).

Figure 6 shows a 1.2 s proton beam cycle with three plateaus. The blue and cyan traces are the field reference and measurement; red shows the current and purple and green show the voltage reference and measurement. The controller runs at 1 kHz, but the regulation acts every 3rd iteration at 333.3 Hz.

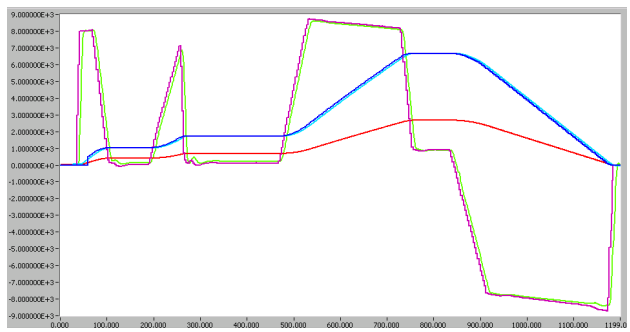


Figure 6: 1.2s cycle with the POPS converter.

AVAILABILITY FOR DOWNLOAD

All the libraries (including libcal) and the fgtest program are freely available under the GNU Lesser General Public License. Visit the website http://cern.ch/cclibs for more information and to download the source. Makefiles for Linux are included.

CONCLUSIONS

These well tested libraries are based on years of operational experience [4] and are a valuable resource for the development of converter regulation software. By centralising these vital components into libraries, the maintenance effort has been reduced and new equipment classes have been developed more quickly.

REFERENCES

[1] Q. King et al, "The All-Digital approach to LHC power converter current control", ICALEPCS'01, THBT004

[2] I.D. Landau, "System Identification and control design", Prentice-Hall International, 1990

[3] F. Bordry, H. Thiesen, "RST Digital Algorithm for controlling the LHC magnet current", CERN/LHC Project report 258

[4] Q. King, "Status of the LHC power converter controls", ICALEPCS'09, MOB003

SIMPLIFIED INSTRUMENT/APPLICATION DEVELOPMENT AND SYSTEM INTEGRATION USING LIBERA BASE SOFTWARE FRAMEWORK

Matej Kenda, Tomaž Beltram, Tomaž Juretič, Borut Repič, Damijan Škvarč, Črt Valentinčič,
Instrumentation Technologies, Solkan, Slovenia

Abstract

Development of many appliances used in scientific environment forces us to face similar challenges, often executed repeatedly. One has to design or integrate hardware components. Support for network and other communications standards needs to be established. Data and signals are processed and dispatched. Interfaces are required to monitor and control the behaviour of the appliances. At Instrumentation Technologies we identified and addressed these issues by creating Libera BASE, which is a framework composed of several reusable building blocks. Libera BASE simplifies some of the tedious tasks and leaves more time to concentrate on real issues of the application. Further more, the end product quality benefits from larger common base of this framework. We present the benefits on examples of instrument implemented on MTCA platform accessible over graphical user interface.

DEVELOPMENT OF ELECTRONIC DEVICES

The need or a vision for a new or improved electronic devices usually emerges from a real-world challenge, issue or a question that needs to be answered.

The device gets defined through the development process in the means of functional (functionality, performance) and non-functional (user experience, scalability, interoperability) requirements. It is then realised through analysis, design implementation, validation and production. This is a day-to-day job in development enterprises all over the world.

For the purpose of accelerator community, the end result of the development is often a measurement instrument, comprising:

- selected hardware architecture (for example Libera hardware architecture A/B, MicroTCA)
- instrument specific electronics, RF
- input signals (analog, digital, timing)
- platform management
- imposed communication protocols
  - PCI, IPMI, Ethernet, SATA, RapidIO
  - Control System protocols, ...
- generic FPGA cores and application specific processing
- application-specific algorithms, parameters, ...

THE ROLE OF SOFTWARE IN RECONFIGURABLE ELECTRONIC DEVICES

The importance of software in electronic devices is increasing. Software is not just an add-on, it became an essential part of electronic devices.

Many of the chips are becoming increasingly integrated and programmable or configurable to some extent. For example FPGA by its nature, ADCs, VCXOs that are controlled over SPI, I2C buses. As a consequence, more software needs to be written to control them.

Software usually acts as the integration layer to make hardware components, application logic and user interfaces to work together. Issues not thought of before the integration need to be resolved then.

Software interfaces are also the points where people communicate with the device. They largely define user experience through graphical, programming and other interfaces. Human behaviour also doesn't comply to standards: any kind of usage of the electronic device needs to be handled in software [1].

REUSABLE SOFTWARE IN MEASUREMENT INSTRUMENTS: SEEING THE PATTERNS

There is a set of concepts that is occurring repeatedly in measurement instruments:

- hardware detection, platform management
- access to functionality implemented in FPGA
- configuration parameters
- notification of changes
- signal acquisition, processing and dispatching
- scaffold for running instrument applications
- supporting standard control system interfaces

Basic Application Support Environment

Powered by Libera BASE

Figure 1: Libera BASE logo.

LIBERA BASE

Libera BASE (Figure 1) is a software framework, developed at Instrumentation Technologies. The design

started in the beginning of 2010 based on many years of previous experience. A common framework was internally needed to develop multiple instruments/applications on the same hardware architecture.

The project's delivery was software framework Libera BASE, which proved itself to be useful widely as initially anticipated [10].

Libera BASE narrows the gap between customer's hardware and the machine's control system. It helps to focus on the application, for which the instrument is designed for with

- software framework for application development and
- intuitive structure and programming interfaces.

Libera BASE simplifies integration into various control systems, however it does not aim to act as a replacement for them.

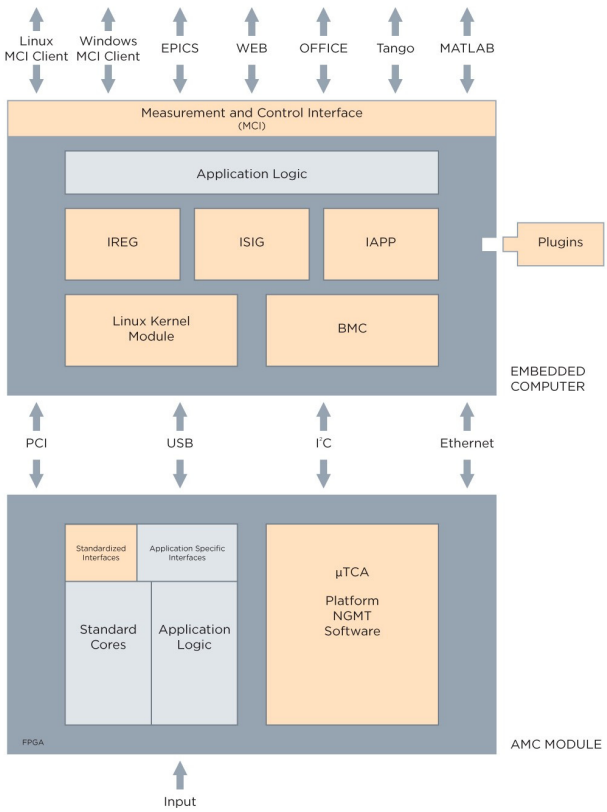


Figure 2: Libera BASE software framework.

Concepts and Building Blocks

Libera BASE consists of several components (Figure 2) that work together in synergy to support applications based on it ([2], [3], [4]).

- **fw**: MicroTCA-compliant platform management
- **bmc**: Hardware abstraction layer (uses IPMI, USB, OpenHPI)
- **lkm**: PCI Linux kernel module relies on a set of standardised FPGA registers
- **ireg**: Application parameters as hierarchical tree
- **isig**: Signal acquisition, processing and dispatching
- **iapp**: Application development framework, plugins
- **mci**: Client programming interface (API) for GNU/Linux and Windows: exposes registry and access to signals
- **tools**: Simple command line tools for automation and scripting
- **adapters**: Matlab and LabView scripts, web, EPICS (EDM, pyEpics) [8], Tango CS [7], FESA [6]

Relation to Libera Instruments

Using Libera BASE to develop instrument application software accelerates development. Measurements of source code size reveal that the share of application-specific software is around 10% of total software; remaining 90% is Libera BASE.

Application software defines the set of user settable parameters, signals, processing and algorithms (Figure 3, Figure 4).

In September 2011, the following instruments use Libera BASE: Libera Brilliance+, Libera Single Pass H, Libera Spectra, Libera LLRF.

Common MCI API is used on all of the instruments which simplifies integration of different instruments into the control system.

Libera BASE is improved incrementally with each new instrument or its new version. Occasionally there are dedicated projects scheduled to develop common functionality.

Improvements of Libera BASE during development of one instrument get incorporated into other instruments on regular basis.

This synergy between the instruments results in better stability and quality of many instruments over time.

```
$ ./libera-ireg dump -h 10.0.3.106 boards.tim2
info
  revision=11650
  health_status=16
clock_info
  adc_frequency=117418690
pll
  locked=true
  clk_good=true
  unlock_thr=100
  vcxo_offset=0
  compensate_offset=false
  pi_coarse
  pi_fine
events
  trigger=61346545890899
  current_time=61346625145926
signals
  pll
    access_type=Stream
    atom_size=64
    group_size=1
    components_names=Err,Dac,Lock,ClkGood,Freq,P,I,IsCoarse
    components_number=8
    components_type=Double
  event
    access_type=Stream
    atom_size=32
    group_size=1
    components_names=id,count,timestamp,reserved
    components_number=4
    components_type=uint64
sc_source=Internal
```

Figure 3: Example of parameters in the registry from Libera Brilliance+.

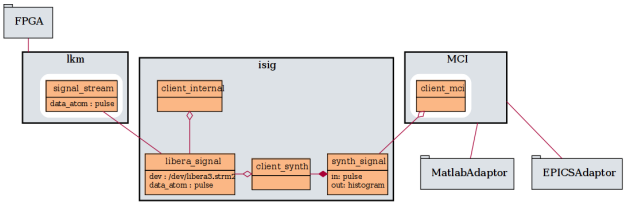


Figure 4: Processing and dispatching a signal.

Libera BASE and Hardware Architectures

Software framework was initially designed to support the modular Libera Hardware Architecture B. Since the coupling between the modules of Libera BASE is low, it is relatively easy to support different hardware architectures with the same software framework. The framework currently supports:

- Libera Hardware Architecture B
- Micro TCA
- Micro TCA for Physics (MTCA.4) [5]

PICMG (PCI Industrial Computer Manufacturers Group [9]) is working on a reference software implementation for applications on MTCA.4. Libera BASE was created sooner and independently from the MTCA.4 reference implementation, because of the market needs.

MTCA.4 reference implementation and Libera BASE share the same mission: help to focus on the application, not the platform.

Libera BASE was presented to PICMG to share ideas and will adapt to the MTCA.4 reference implementation when available.

Libera BASE and Libera Platform A

Instruments based on Libera Platform A provide CSPI API for integration with control systems.

CSPI is only an API, however Libera BASE is a complete framework for application development.

CSPI was designed for a single hardware architecture and instrument. Libera BASE can be adapted to support Libera hardware architecture A, but the opposite would be harder.

Future Plans

Feature set of Libera BASE is becoming rich enough to be able to implement many applications on different hardware architectures.

In the next period it is intended to improve out-of-the box connectivity, abilities to customise/extend the applications and usability.

USAGE SCENARIOS OF LIBERA BASE

Development of New Instruments

Libera BASE software framework provides a productive environment for development of a new instrument.

Integration of Instruments into their Working Environment

The MCI programming interface is designed to be simple, intuitive and powerful. Different instruments expose the same networked API, which can be used from GNU/Linux and Windows clients (Figure 6).

The API is used to implement different client programs (command line, graphical) and adaptors for integration with applications and control systems.

Different instruments have already been accessed from the following clients and adaptors:

- graphical user interface
- command line interface
- Matlab, SCILab
- OpenOffice.org spreadsheet
- mobile devices: iPhone, iPad, Android-based phones (Figure 5)
- web browsers
- EPICS EDM

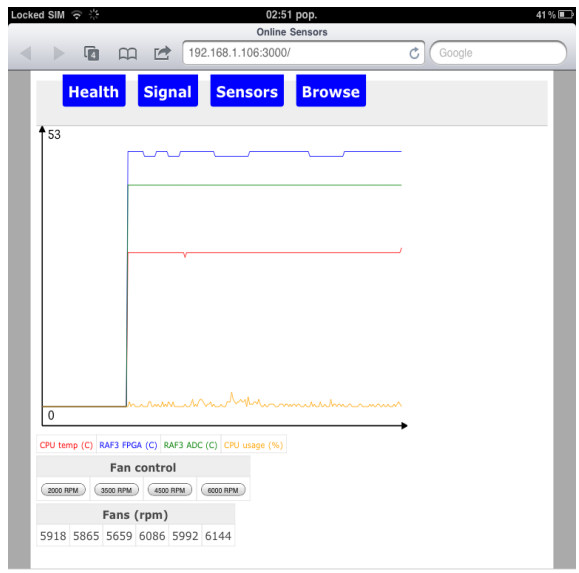


Figure 5: Controlling an instrument from a mobile device.

Customising and Extending an Instrument

Several tools are provided to customise Libera instruments:

- FPGA development kit (FDK)
- FPGA to registry map (no programming needed)
- Software plugins

New parameters and signals implemented when extending the instrument are exported through MCI API in the same way as those originally provided by the instrument.

Source Code Example

```
#include <iostream>
#include "mci/mci.h"
#include "mci/node.h"

int main(int a_argc, char* a_argv[])
{
    mci::Init(a_argc, a_argv);

    auto root = mci::Connect("192.168.1.100", mci::Root::Application);

    //
    // Dump complete tree of registry parameters
    //
    auto nodes = mci::SubTree(root);
    for (auto i(nodes.begin()); i != nodes.end(); ++i) {
        std::cout
            << i->GetFullPath() << " = "
            << i->ToString() << std::endl;
    }

    //
    // Access and modify a parameter
    //
    mci::Path path = mci::Tokenize(
        "boards.raf3.tbt.spike_removal.averaging_window");

    auto n = root.GetNode(path);

    // Read a numeric parameter
    int32_t aw = n;
    std::cout << "Averaging window: " << aw << std::endl;

    // Modify a numeric parameter
    aw = 16;
    n = aw;

    mci::Shutdown();
}
```

Figure 6: Example of C++ source code.

SUMMARY

Libera BASE narrows the gap between your hardware and the machine control system. It simplifies development of instruments and integration into control systems.

Libera BASE provides MCI programming interface that is powerful, easy to learn and use.

The software framework is designed for various hardware technologies.

High level of re-usability increases reliability and quality of instruments.

Supports reconfigurable instruments with its modular structure and extendibility.

REFERENCES

[1] Matej Kenda, Hinko Kočevar, Tomaž Beltram, Aleš Bardorfer , “Programming Interfaces for Reconfigurable Instruments ”, PCaPAC 2010, Saskatoon, WEPL032, October 2010.

[2] Gamma, Helm, Johnson, Vlissides, “Design Patterns”, Addison-Wesley, 1995.

[3] http://en.wikipedia.org/wiki/Software_design

[4] http://en.wikipedia.org/wiki/Software_framework

[5] “Specification MTCA.0”, PICMG, July 2006

[6] T. Hoffmann, GSI, “FESA - The Front-end Software Architecture at FAIR”, PCaPAC 2008, Ljubljana, WEP007, October 2008.

[7] www.tango-controls.org

[8] www.aps.anl.gov/epics/

[9] www.picmg.org

[10] Instrumentation Technologies, “Libera BASE home page”, www.i-tech.si/accelerators-instrumentation/technologies/libera-base.

CONTROLLING THE EXCALIBUR DETECTOR

J. A. Thompson, I. Horswell, J. Marchal, U. K. Pedersen, Diamond Light Source, Oxfordshire, UK.
S. Burge, J. D. Lipp, T. Nicholls, Science and Technology Facilities Council, Oxfordshire, UK.

Abstract

EXCALIBUR is an advanced photon counting detector being designed and built by a collaboration of Diamond and the STFC. It is based around 48 CERN Medipix3 chips arranged as an 8 x 6 array. The main problem addressed by the design of the hardware and software is the uninterrupted collection and safe storage of image data at rates up to one hundred 2048 x 1536 frames per second. This is achieved by splitting the image into 6 'stripes' and providing a parallel data path for each stripe all the way from the detector chips to the storage. This architecture requires the software to control the configuration of the stripes in a consistent manner and to keep track of the data so that the stripes can be subsequently stitched together into frames.

INTRODUCTION

A 2D position sensitive detector is required for use in a range of imaging experiments at Diamond Light Source. Initially the detector will be applied on photon beam line I13, but the resulting design may be applied on other beam lines and in other applications.

The primary application will be in the technique of Coherent Diffraction Imaging (CDI) [1]. The detector will also find applications in Photon-correlation Spectroscopy (XPCS) [2], full-field microscopy and holo-tomography.

The Medipix3 Device

The Medipix3 device is a 256 x 256 pixel photon counting detector. It has three basic operating modes, single-pixel, charge-summing and colour. The EXCALIBUR instrument will initially support single-pixel and charge-summing modes.

The charge that is collected by a pixel due to an impinging photon is measured and compared to a programmable threshold. Only charge pulses that meet the threshold's requirements are registered by the 12 bit counter. There are two 12 bit counters associated with each pixel. These are normally used so that one is counting photons while the other is being read out, to eliminate dead time between frames.

Targeted Capabilities

Table 1 summarises the capabilities of the EXCALIBUR detector and draws comparison with the Pilatus [3].II and XFS detectors.

Table 1: Specification And Comparison With Competitors

Parameter	EXCALIBUR	Competitors
Frame size	2k x 1.5k pixels	
Maximum frame rate	100Hz, 12 bits continuous 1kHz, 12 bits burst 30kHz, 1 bit histogrammed	Pilatus II < 300Hz Pilatus XFS > 10kHz
Dead time between frames	0	Pilatus II: ~2ms defined by chip read out time.
Pixel size	55 um	Pilatus II 172um Pilatus XFS 75um
Dynamic range	12 bits	Pilatus XFS 12 bits
Quantum efficiency	~50% at 15keV	~50% at 15keV

HARDWARE ARCHITECTURE

The hardware architecture is shown in Fig. 1. It consists of a sensor array, a number of front-end modules (FEMs) and a cluster of read-out node PCs.

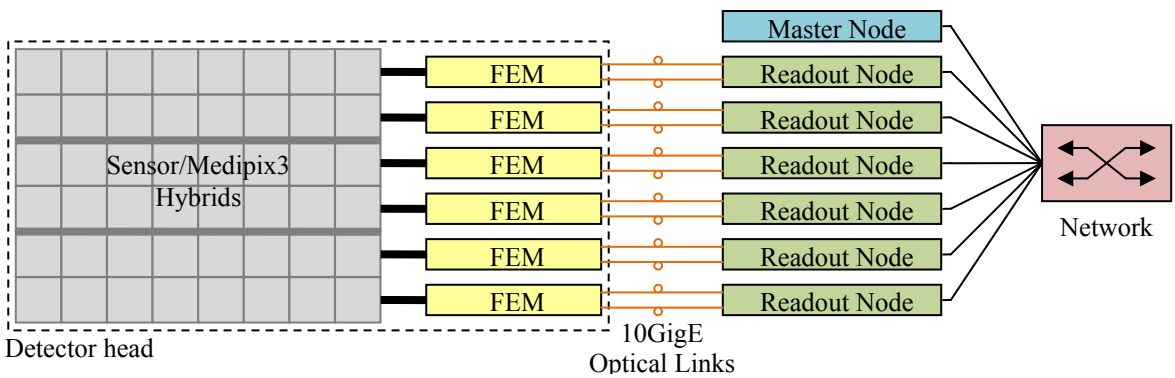


Figure 1: Hardware architecture.

Sensor Assemblies

The sensor assembly consists of three hybrid modules, each containing an 8 by 2 array of sensors and Medipix3 chips. The detector assembly consists of three hybrid modules, each containing a large silicon sensor bonded to an array of 8 x 2 Medipix 3 chips. The gaps between the chips on one module are 3 pixels (the minimum possible); the sensor pixels that cover these gaps are larger, as shown in Fig 5. A 124 pixel wide inactive region is present between modules due to the presence of wire bond pads connecting the chips to their read-out electronics.

Front End Modules

A FEM is provided for each horizontal row of sensors, giving a total of 6 FEMs. Each FEM co-ordinates the acquisition of data by a row of Medipix3 chips and also provides access to most of the registers of the 8 Medipix3 chips for the embedded software. From the point of view of the software, a FEM provides an image 'stripe' that is 2069 by 256 pixels. The seven inter-chip gaps (each of 3 pixels) are included in the output data but do not contain valid data.

Computing Resource

Six read-out nodes and one master node make up the computing cluster for EXCALIBUR. The read-out nodes communicate with the FEMs through 10G Ethernet fibre optic links. Connection to the Diamond Light Source network backbone is then made through six 1G Ethernet links to a switch with a 10G upstream connection.

Data Storage

The primary storage for data captured by EXCALIBUR will be a Lustre [4] distributed file system. Files may also be stored locally on the read-out node's file systems if required.

EMBEDDED SOFTWARE ARCHITECTURE

The embedded software is based on EPICS [5] using the area detector module [6]. Figure 2 shows an overview of the data path software; the classes ADDriver, NDPluginDriver and NDPluginFile are area detector base classes supplied by the module library.

Image Stripe Handling

The AdFem class is responsible for interfacing to the FEM. It receives image stripes, places them into standard area detector buffers and then passes them on for further processing. In addition, it provides EPICS process variables that allow the control of acquisition and the configuration of the Medipix3 and FEM devices.

The pixel arrangement in the stripes is different for each of the pair of FEMs that service a sensor module, due to the rotation of the Medipix3 chips. This is corrected by the AdFemFix class. This function is provided as a separate area detector plug-in to allow it to run on a different core to the FEM readout, keeping the throughput high.

Summary Image Production

The AdSlaves and AdMaster classes together provide the mechanism for transferring at a low rate (configurable

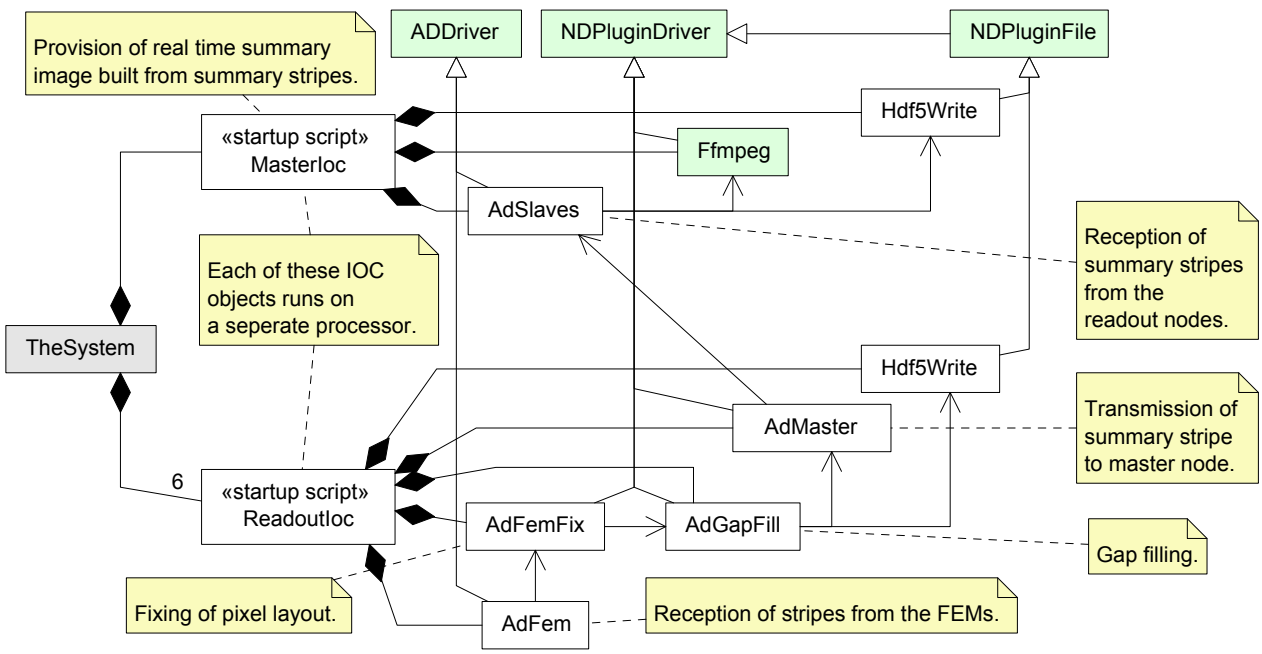


Figure 2: UML class diagram showing a summary of the data path software.

as 1 in N) a summary image stream from the read-out nodes to the master node. A class diagram is shown in Fig. 3.

The AdMaster class transfers every Nth stripe to the AdSlaves class through a TCP socket. The stripes are written into the correct place in a single area detector buffer to stitch them together. Once a stripe is received from each read out node, the complete frame is passed on for further processing on the master node. This will normally include an MPEG streaming plug-in to provide data for a suitable viewer.

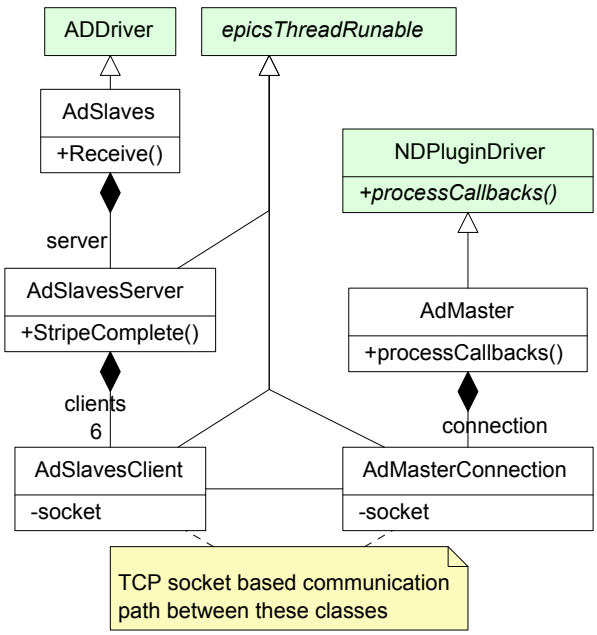


Figure 3: UML class diagram of the summary image processing.

Parallel File Handling

The read-out nodes each open the same file on the Lustre file system and write to the correct part of the file to assemble the stripes into the frame. Not only does this avoid processing by the instrument; it also utilises the multiple parallel write stream capability of Lustre to keep the data rate high.

The files are written in the HDF5 format [7]. The file writing plug-in utilises the version of the HDF5 software that uses the openMPI parallel processing library [8].

Filling the Gaps

The gaps between the adjacent Medipix3 chips are required to be filled, either by a constant value or by interpolation between the pixels surrounding the gap. A class diagram of the plug-in responsible for this task, AdGapFill, is shown in Fig. 4. The chip layout and gap details are shown in Fig. 5.

The large gaps between the modules are always filled with a constant value. The HDF5 file format provides the ability to fill automatically gaps in the recorded data with a constant. Advantage is taken of this feature to fill the large gaps without increasing the data rate of the system.

The small gaps may be filled with a constant value or interpolated, according to configuration. The small vertical gap pixels are already present in the stripe data, so the gap filling plug-in writes the constant or interpolated value as appropriate. The gap filling feature of HDF5 is used again for the horizontal gaps in constant fill mode.

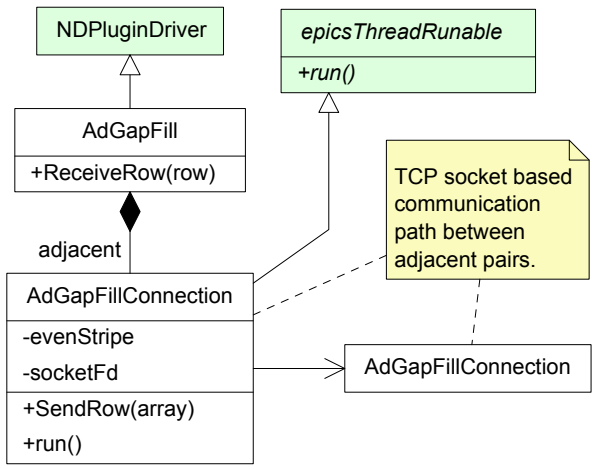


Figure 4: UML class diagram of the gap filling processing.

Interpolating across the small gap between stripes is not straightforward; it requires data communication between the adjacent gap filling plug-ins. The plug-in on the upper side of the gap receives the top row of pixels from the plug-in on the lower side. It then generates extra rows of interpolated pixels which extend the stripe downwards, over the gap.

The sensor areas of the pixels adjacent to the small gaps extend over the gap, as shown in the right of Fig 4. This means that photons landing in the pixel gaps are still captured. Interpolation is therefore concerned with sharing captured photons between the edge pixels and the gap pixels.

Configuration Control

The read-out nodes each provide EPICS process variables that control the operation of a single FEM and its associated Medipix3 chips. These are all available to allow the individual control necessary during many engineering operations, especially during calibration.

For normal user operation, co-ordinated control of the entire instrument is required. This is provided by a separate set of EPICS process variables residing on the master node that model the instrument in a user-oriented manner. These process variables are then expanded and distributed to the read-out nodes by the configuration control state machine. The same entity also monitors the operating state of the whole instrument and audits the distributed configuration.

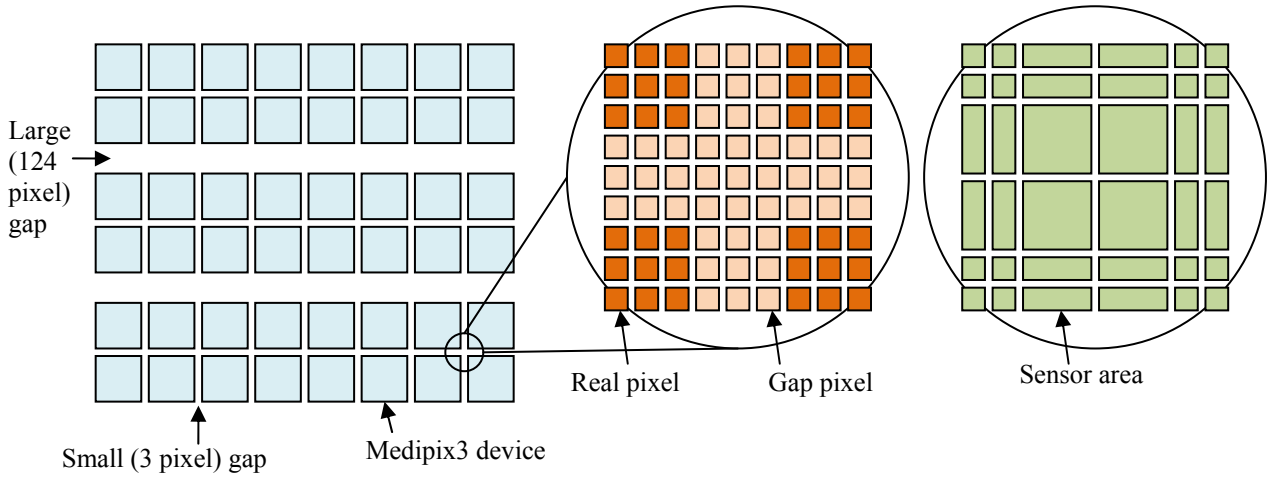


Figure 5. Medipix3 device layout and gap details.

Instrument Calibration

There are many calibration procedures to be undertaken on the instrument to provide the necessary information for correct operation. These include pixel threshold trimming, flat field correction and bad pixel maps. In general, the procedures will be performed by external scripts and will result in calibration files that are loaded onto the instrument and activated at the appropriate time.

SOFTWARE TESTING

The testing of the software was carried out in two phases. The first phase used a software simulation of the hardware with an extensive set of automatic test routines. This provided an automatic record of the testing undertaken and an easy way of repeating the tests at any stage during instrument development.

The second phase was a period of integration and testing with the instrument hardware. This necessarily involved rather more manual intervention and grew as the various parts of the instrument were brought together.

Instrument Simulator

To allow unit testing of the embedded software before the hardware was available, a simulation of the hardware was produced. This consists of three parts, the Medipix3 chip simulation, a FEM simulation and a back door through which the test suite can control the simulation.

The Medipix3 chip simulation is reasonably detailed from a software point of view. It includes the ability to generate pixel data that responds to the settings of the chip's registers in a reasonably appropriate manner.

The FEM simulation is rather simpler. Its main job is getting data into and out of the Medipix3 chips, so many of its functions are invisible to the software.

The simulation connects to the embedded software at a point where the FEM driver library connects to the embedded software proper. This allows the simulation to take the form of a library that is linked in place of the FEM driver library.

Automatic Test Suite

The tests written covered low level Medipix3 register access, configuration management and image frame processing. They were written in Python [9] utilising the PyUnit standard library. Cases are able to control the simulation through its backdoor, allowing both valid and fault conditions to be tested.

CONCLUSIONS

An instrument capable of sustained capture of 2k x 1.5k pixel images at 100 frames per second has been built. This corresponds to a throughput of just over 5Gbps to the Lustre file system in 6 coordinated, parallel data streams.

REFERENCES

[1] http://en.wikipedia.org/wiki/Coherent_diffraction_imaging
[2] http://sector7.xor.aps.anl.gov/~dufresne/UofM/xpcs.html
[3] http://www.dectris.ch/
[4] http://wiki.lustre.org/index.php/Main_Page
[5] http://www.aps.anl.gov/epics/about.php
[6] http://cars9.uchicago.edu/software/epics/areaDetector.html
[7] http://www.hdfgroup.org/HDF5/
[8] http://www.open-mpi.org/
[9] http://www.python.org/doc/

PC/104 ASYN DRIVERS AT JEFFERSON LAB

J. Yan, T. Allison, S. Witherspoon
Jefferson Lab, Newport News, VA 23606, U.S.A.

Abstract

Asyn Driver was applied for PC/104 IOC serial communication systems at Jefferson Lab. We chose the ines GPIB-PC/104-XL as the GPIB interface module and developed a low lever device driver that is compatible with the asynDriver. Instrument device support was created to provide access to the operating parameters of GPIB devices. A Low level device driver for the serial communication board Model 104-COM-8SM was also developed to run under asynDriver. This serial interface board contains eight independent ports and provides effective RS-485, RS-422 and RS-232 multipoint communication. StreamDevice protocols were applied for the serial communications. The asynDriver in PC/104 IOC applications provides a standard interface between the high level device support and hardware level device drivers. This makes it easy to develop the GPIB and serial communication applications for PC/104 IOCs.

INTRODUCTION

PC/104 embedded IOCs that run RTEMS and EPICS have been applied in many new projects to control all kinds of different devices in Accelerators at Jefferson Lab [1]. Different commercial PC/104 I/O modules on the market such as digital I/O, data acquisition, and communication modules are integrated in our control system. Many devices and instruments are controlled via serial and GPIB communications. With the availability of low cost PC/104 I/O modules, it would be easy to configure these systems. However, device drivers and device support for these new PC/104 I/O modules have to be written under RTEMS and integrated in EPICS system. Solutions that apply EPICS tools and develop the generic drivers for these new modules were researched. asynDriver[2][3] is a general purpose facility for interfacing device specific code to low level communication drivers. It provides a structured environment for developing support for hardware devices both asynchronous and synchronous communications. Since asynDriver provides the EPICS IOC device support, only low-level device drivers for the new I/O modules need to be programmed. This paper presents the selection of PC/104 I/O modules for serial and GPIB communications, the programming of low-level device drivers for these modules, and some samples of applications.

HARDWARE CONFIGURATIONS

The Kontron PC/104 processor board provides two ports, Com1 and Com2, for serial communication. When RTEMS is running, Com1 is configured as the console and Com2 is disabled. Some applications may need RS-232 serial communication, so Com2 can be enabled and configured to provide this function. For some other applications, one PC/104 IOC was required to provide multiple-port serial communications. Therefore, a serial communication board, Model 104-com-8SM, was chosen for this purpose. This serial interface board contains eight independent ports and provides effective RS-485, RS-422 and RS-232 multipoint communication. By setting jumpers on the board, each channel can be configured to any of these modes. A type of XR16L788 octal Universal Asynchronous Receiver and Transmitter (UART) on the board is used as the Asynchronous Communications Element (ACE). A XILINX XCR3256XL Complex Programmable Logic Device (CPLD) is applied to control the XR16L788 chip and communicate with PC/104 ISA bus. Two 40-pin connectors are used for interfacing to communication lines for 8 channels. Figure 1 shows the eight-channel serial communication chassis, where the 104-com-8SM board is stacked on a PC/104 processor module.

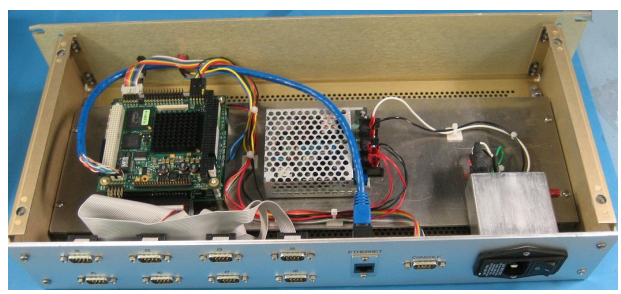


Figure 1: 8-Channel Serial Communication Chassis.

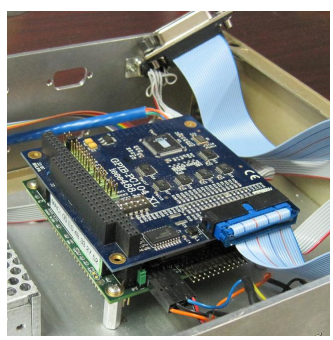


Figure 2: GPIB-PC/104-XL Module Stacked on the PC/104 IOC.

The ines GPIB-PC/104-XL was chosen as the GPIB interface control module for the GIPB communication system. This module has an iGPIB 72110 chip, which provides an interface between a microprocessor system and the GPIB specified in the IEEE Std. 488.1-1087 and 499.2-1987. The iGPIB 72110, a 100-pin TQFP package, is register compatible with NEC uPD7210 in GPIB

Talker/Listener applications. Figure 2 shows the GPIB-PC/104-XL module stacked on a PC/104 processor module through the ISA bus. A 24-pin ribbon cable provides the GPIB interconnects between the interface module and up to 15 GPIB devices.

SOFTWARE DRIVERS

Since the asynDriver provides all components of device support, only the low-level driver needed to be programmed for the new I/O hardware. We wrote drivers for the ines GPIB-PC/104-XL module, 104-com-8SM model, and on-board COM2 port.

On the 104-com-8SM serial module, XR16L788 integrates functions of 8 enhanced 16550 UARTs, a general purpose 16–bit timer/counter and an on-chip oscillator. Device configuration registers include a set of four consecutive interrupt source registers that provides interrupts-status for all 8 UARTs, timer/counter and a sleep wake up indicator. Each UART channel has its own 16550 UART compatible configuration register set for individual channel control, status, and data transfer. The driver was written based on the source code for drvAsynSerialPort.c in the asyn package. However, the register access and interrupt handler are very different between the PC-104/RTEMS and the targeted VME/VxWorks. The structure of the driver can be described as the following:

- Create the structure for ports hardware-specific information.
- Initialize each port.
- Create the IRQ handler for each port
- Create and register asyn port name for each port.
- Set baud rate, parity, bits, stop bit for each serial port.
- Link with higher level routines of asynDriver.

The iGPIB 72110 chip on the GPIB interface module is register-compatible with NEC uPD7210, so the GPIB control interface driver was developed under the specification of uPD7210 registers. The uPD7210 is an intelligent control designed to provide high-level protocol management of the GPIB communication. Control of the uPD7210 is accomplished via 16 internal registers. The driver was programmed based on the file of drvNi1014.c in the asyn package. Here is the structure of the driver:

- Create the port name.
- Set the base I/O address.
- Set the interrupt vector and level.
- Create RTEMS IRQ handler.
- Register the asyn port and connect it with asyn.

All the low-level drivers were compiled as shared library files, so any application can call them.

APPLICATIONS

Most function generators and measurement instrument devices have a GPIB connector. In our new RF control system test stand we have three devices that need to be controlled via GPIB. These devices are CG635 Synthesized clock generator, Agilent E4428C Signal Generator, and Giga-tronics 8540C Universal Power Meter. Figure 3 shows the schematic of the RF control system test stand. For each GPIB device we developed device support according the protocol of asynGpib device support. First of all, we have to determine the set of operations that each device will have to perform. Then we create the device support file for each device and declare the command array. Each command array entry describes the details of a single I/O operation type. Finally, the application database uses the index of the entry in the command array of each file to provide the link between the process variable and the device I/O operation to read or write the value. In this application, CG635 and E4428C mostly require write operation, while 8540C power meters employ reads.

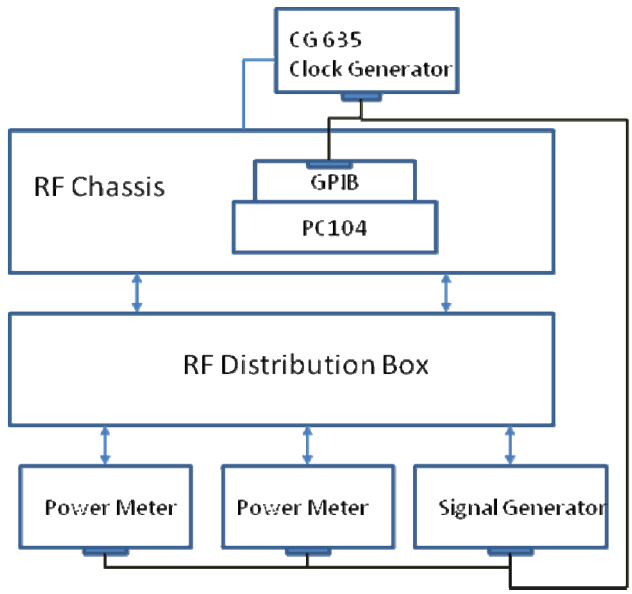


Figure 3: GPIB Communication for RF Control System Test Stand.

Beam Position Monitor (BPM) test stand is a system that calibrates 4-wire BPM cans. This system consists of a 4–channel BPM data acquisition board [4], PC/104 IOC and serial module, two Applied Motion STAC6-Si stepper drives, and a BPM can attached with two HT23-549 stepper motors. The data acquisition board reads the electrical signals from the four (x+, x-, y+, y-) electrodes inside the BPM can. Each stepper motor controls the movement of the BPM can in horizontal (X) or vertical (Y) directions. When the BPM can is moved to a specific position, two serial ports read back the value of the position and the data acquisition board would simultaneously sample the electrical signals from four antennas inside the BPM can. Stepper motors are controlled by writing and reading a set of commands to the stepper drivers. *StreamDevice* [5] is applied as the device support for controlling stepper drivers.

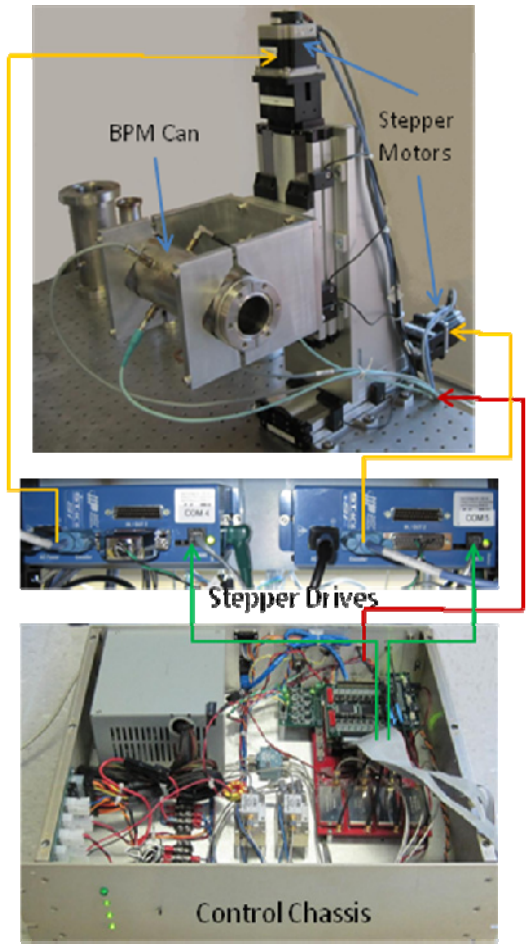


Figure 4: BPM Test Stand Stepper Motor Control System.

StreamDevice is a generic EPICS device support for devices with a "byte stream" based communication interface, such as RS-232, RS-485, GPIB, and telnet-like TCP/IP. It can be configured for any device type with protocol files in plain ASCII text which describes the commands a device understands and the replies it sends. In this application, each stepper drive has one specific protocol file that defines the commands to communicate between the device and database records. Each record with *StreamDevice* support runs a command from the protocol file to read or write its value. The advantages of *StreamDevice* are the ease to configure commands to run devices and the support for all standard record types in EPICS base which have device support.

CONCLUSIONS

Asyn driver has been applied as generic device support for the embedded PC/104 IOCs that run RTEMS and EPICS. The low level drivers for a GPIB interface module and 8-channel serial communication module were developed to be compatible with the asyn toolkit. A number of applications have been written to use the PC/104 asyn driver in our control system at Jefferson Lab. The asyn driver provided an easy integrated solution for PC/104 serial communication applications.

ACKNOWLEDGEMENT

Thanks to the Instrument and Control group and Software group in Jefferson Lab for their technical support.

REFERENCES

[1] J. Yan, etc. "PC/104 Embedded IOCS at Jefferson Lab", Proceedings of ICALEPCS2009, Kobe, Japan, P. 230-233.

[2] M. R. Kraimer, etc. "EPICS: Asynchronous Driver Support", 10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems, Geneva, 10-14 Oct 2005 PO2.074-5.

[3] http://www.aps.anl.gov/epics/modules/soft/asyn/R4-16/asynDriver.pdf.

[4] J. Musson, etc. "Reduction of Systematic Errors in Diagnostic Receivers Through the Use of Balanced Dicke-Switching and Y-Factor Noise Calibrations", Proceedings of PAC09, Vancouver, BC, Canada, P.4057-4059.

[5] http://www-group.slac.stanford.edu/cdsoft/icalepcs01/ StreamDevice.pdf

RECENT DEVELOPMENTS IN SYNCHRONISED MOTION CONTROL AT DIAMOND LIGHT SOURCE

B. Nutter, T. Cobb, M. Pearson, N. Rees, F. Yuan, Diamond Light Source, Oxfordshire, UK

Abstract

At Diamond Light Source the Experimental Physics and Industrial Control System (EPICS) [1] is used with a variety of motion controllers. The use of EPICS ensures a common interface over a range of motorised applications. We have developed a system to enable the use of the same interface for synchronised motion over multiple axes using the Delta Tau Programmable Multi Axis Controller (PMAC). Details of this work will be presented, along with an example and possible future developments.

INTRODUCTION

Diamond Light Source is host to around twenty operational X-ray beamlines, each of which has various motorised devices in order to control and focus X-ray photons. These devices range from simple diagnostics and attenuators, to mirrors and double crystal monochromators, and complex multi-axis diffractometers and hexapod systems. There can be well over one hundred motors on a photon beamline, from simple stepper motors to servo motors, piezo motors and nano motors.

The control of motors is accomplished by using dedicated motor controllers that typically operate a servo control loop and amplifier for each axis. On top of this we have motor control software that sends commands to the controllers and reads back status information. At the highest level we make use of the EPICS control system and the Generic Data Acquisition (GDA) scientific framework [2] to provide a consistent user interface to each type of motor.

In addition to the standard control of each axis we perform synchronised motion on multiple axes. We do this at various levels, such as sending simultaneous move commands from the high level software, or defining a co-ordinate system at the motor controller level (which lets us achieve true synchronised motion).

MOTION CONTROL HARDWARE IN USE AT DIAMOND

Diamond Phase I beamlines use the Delta Tau Turbo PMAC 2 VME Ultralight motor controller. This is a 32-axis controller which sits inside a VME crate alongside a CPU card running VxWorks and EPICS. The Ultralight controller uses a fibre optic communication link to two 16-axis Universal Motion and Automation Controller (UMAC) units, which in turn interface to limit switches, encoders and external amplifier crates.

At the end of Phase I Diamond underwent a tender process for Phase II beamline motion controllers. It was determined that the use of Ethernet-based controllers would be a more flexible and economic alternative than the traditional VME-based controller. The controller we chose was the Delta Tau Geobrick Low Voltage Integrated Motion System (LV IMS). This is a Turbo PMAC 2 controller in a 4U enclosure, with an integrated amplifier board. The amplifier board has 4 or 8 axes of output, each of which can be configured in software to be either a stepper or a servo amplifier. This made it possible not only to control DC stepper motors, with various encoder feedback mechanisms, but also to control brushless and DC servo motors. Step and direction outputs are also available to control pico motors and ultra-high resolution external stepper amplifiers.

The product has suffered somewhat from early design teething troubles, mainly concerning the amplifier firmware. However, all solutions to date have been compatible with the existing hardware platform. We are also seeing an increasing number of very small motors, typically below 500mA, and the current resolution of the Geobrick amplifiers (which are rated at 5A continuous, 15A peak, but have a zero current set point accuracy of nearly 100mA) is becoming an issue.

Diamond has a number of other motor controllers on site. These include the Newport XPS controller, mainly used on Newport diffractometers, and a small number of Aerotech Ensemble controllers, required to take advantage of the smoothness and precision of their inbuilt linear amplifier stages. We also have many piezo stack control devices from Queensgate, PI and Jena, which are controlled by either analogue signals or RS232.

EPICS MOTOR CONTROL SOFTWARE

For each specific motor controller there is a software driver based on the Asyn software framework [3]. Asyn is a low-level driver software abstraction framework, providing synchronous and asynchronous commands and responses. The drivers are either developed in house (normally building on the work of similar drivers) or use is made of drivers developed elsewhere. Above these drivers is a consistent user interface in the form of an EPICS record, called the Motor Record. This provides the user with functions such as *move*, *stop*, *home*, etc., as well as support for modifying the motion properties such as velocity or acceleration. The EPICS Motor Record is documented in full at [4].

The Asyn-based motor driver framework that sits underneath the EPICS motor record has been developed by Diamond and the Advanced Photon Source, and has been previously been presented [5]. The advantages that the driver framework gives us are mainly the ease of adding new drivers for new controllers, the ability to use multiple EPICS records to control a motor and its

parameters and the ability to monitor a motor status asynchronously (by using an event driven polling thread).

GDA SOFTWARE INTERFACE TO EPICS MOTOR SOFTWARE

The GDA is a data acquisition system developed at Diamond to provide a user interface and experiment platform for Diamond beamline users. It has a client-server architecture and includes support for the EPICS channel access protocol. A Java implementation of the EPICS channel access protocol, called Channel Access for Java (CAJ) [6], is used to provide both synchronous and asynchronous communication with EPICS. When configuring the GDA for a beamline, the EPICS motor record name is obtained dynamically from an EPICS-GDA interface XML file, generated during EPICS build and deployment, to ensure consistency between the two systems.

Motor objects in GDA are designed to be in state synchronisation with the underlying EPICS motor record to support the simultaneous use of GDA and EDM (an EPICS-aware Motif-based graphical toolkit called Extensible Display Manager [7]). Access control of EPICS motors is also supported, and can be used to restrict access to particular devices such as insertion devices or front end motors.

Beamline users normally collect data and access EPICS motors using the GDA scan mechanism. This is a very flexible framework capable of capturing a complete data set in experiments, including metadata such as beam conditions and sample environments. It allows users to move motors in multiple dimensions, typically from a start position to an end position in predefined steps, and to collect arbitrary data during a scan.

COORDINATED MOTION SOFTWARE

We have been able to achieve co-ordinated multi axis motion in several ways:

- Sending simultaneous move commands to multiple axes at a high level, using GDA to control multiple motor records. An example of this is slit scanning, where the GDA performs its own calculations for slit gap and centre positions. This is achieved by creating a GDA object to represent the combined motor axes. The object can then be used by the GDA scan mechanism.
- Sending simultaneous move commands to multiple axes at the EPICS level. A set of EPICS records work together to calculate combined motion and send the move commands one-by-one to the motor records. This is similar to the GDA-based mechanism, except in this case the GDA only knows about one EPICS motor record, for example a slit gap 'motor'.
- Using 'deferred moves'. This is a technique that involves the driver queuing up move commands and sending them all at once to the motor controller. This technique is useful if one is attempting to move multiple axes that are 'grouped' together on the motor controller itself. When axes are grouped together, an axis can go into the 'moving' state when we are actually only moving a different axis. This causes an issue when using the EPICS motor record and attempting to drive two axes by sending separate commands, because, to avoid ambiguity, EPICS needs the motor to be idle before the move starts if the client also wants to be notified asynchronously of the move completion. In effect, the motor record does not permit a move to be sent while an axis is already in motion (unless the previous move is stopped first). Deferred moves are a way around this issue, and provide a greater degree of move synchronisation than can be achieved by using high level software alone.
- Making use of the coordinated motion capabilities of the controller, and defining a 'coordinate system' for multiple axes.

We will expand on the 4th option. On the PMAC controller, this method involves setting up the axes in a co-ordinate system, which means defining forward and inverse kinematic calculations and defining a PMAC PLC program to provide combined axis position read-back. An example for a pair of slits is shown in Figure 1.

```
&2
#1->I
#2->I
OPEN FORWARD
CLEAR
Q1=(P1+P2)/2
Q2=P1-P2
CLOSE

OPEN INVERSE
CLEAR
P1=Q1+Q2/2
P2=Q1-Q2/2
CLOSE
```

Figure 1: PMAC forward and inverse kinematics.

In Figure 1 the notation &2 defines co-ordinate system number 2, and the #1->I means that axis 1 will be placed in that co-ordinate system. The Q1 and Q2 variables are from the PMAC 'Q-variable' range on the PMAC which have co-ordinate system scope. Here we are using Q1 and Q2 to represent the combined motion. P1 and P2 are used to represent the real axis position, based on the forward kinematics. We also need to define a PLC program to calculate the real combined motion positions, and this is used for position read-back by the EPICS driver.

```
OPEN PLC 2
CLEAR
Q81= (m162 + m262)/2  ;real center
Q82= (m162 - m262)    ;real gap size
CLOSE
```

Figure 2: Position readback PMAC PLC program.

In Figure 2 we are using Q81 and Q82 to hold the combined motion positions, calculated from the actual motor positions held by the PMAC (which are the mX62 variable, where the X is the axis number). In the above examples we have omitted scale factors, which are required to convert to and from engineering units and PMAC position 'counts'. We need to define a PLC program such as this in order to provide real combined axis position read-back, because the P1 and P2 variables are specific to the kinematic buffer.

Finally, there is also a PMAC motion program, which is executed whenever a combined motion command is sent to the controller. The motion program then makes use of the co-ordinate system, which we defined above.

The EPICS driver for the PMAC has been modified to support the functionality we have described above. There is a coordinate system driver that writes combined motion demand values into Q-variables on the PMAC and executes the motion program on the PMAC. It also reads the Q-variables from the PLC program in order to obtain the position for the combined axes.

This is implemented in the EPICS co-ordinate driver in the motorAxisMove function. It is accomplished by sending three commands to the controller:

- Write new demand values into the appropriate Q-variables for the coordinate system
- Abort the current move (to ensure that the axes are enabled)
- Run the motion program

In the co-ordinate system driver there is a separate thread which polls the actual combined motion axis positions (provided by the PLC program described above). This provides continuous position updates to the motor record, at a configurable rate (typically 10 or 20Hz).

The EPICS coordinate system driver fits into the same asyn motor framework as the existing drivers, and so can be used with the same EPICS motor record code. This means that we have a consistent user interface to both real axes and combined coordinated axes. The PMAC co-ordinate system driver is packaged with the tpmac EPICS support module [8], or is available stand-alone as the pmacAsynCoord module.

DIAMOND STANDARDS FOR DELTA TAU PMAC DEVELOPMENT

The coordinated PMAC driver does make some assumptions about how the PMAC co-ordinate system has been set up. For example, it assumes that the PLC read-back positions will be stored in the variables Q81 to Q89. In order to ensure that PMAC software development is kept in sync with the EPICS driver we developed a set of PMAC programming standards, both for co-ordinate systems and general purpose PMAC PLC programs. These programming guidelines have been written down and we routinely forward them onto equipment suppliers who develop PMAC motion control systems for us. They also serve as a useful guide within the group when we develop new PMAC software. We hope that these standards will be useful to other EPICS sites that are using PMAC controllers, as this will ensure the PMAC co-ordinate system driver will be applicable outside Diamond.

A selection of the standards we have developed is described here. First, we list the reserved PLC program numbers and their intended use in Fig.3.

PLC1	initialization routines
PLC2	motion stop detection
PLC3	amplifier enable routine
PLC4	encoder loss detection
PLC5	CPU use reporting
PLC6	amplifier setup
PLC7	auto amplifier power off

Figure 3: Reserved PLC numbers and their function.

PLCs 8 to 16 are free for application-specific code and PLCs 17 to 31 are reserved for co-ordinate system position reporting (as described in this paper). In order to avoid conflicts PLCxx (where xx is the PLC number) should only use PMAC P-variables (which have global scope) in the Pxx00 to Pxx99 range.

Motion programs and co-ordinate systems should use Q-variables where possible, and only use P-variables in the range P3200 to P4200 (to avoid conflicting with PLC programs). For co-ordinate systems we reserve the following Q-variable ranges:

- Q71 to Q79 – co-ordinate system demand positions.
- Q81 to Q89 – co-ordinate system readback positions (these are the positions calculated by the position reporting PLC program).
- Q400 to Q439 – backlash compensation settings in coordinate systems.

In addition to the above, Delta Tau reserve various variable ranges for their own use. We also document various PMAC programming guidelines, such as best practice for developing a homing PLC program. For example it is bad practice to disable limit switches during a homing program, in case the program fails and therefore fails to re-enable the limit switches.

CONCLUSION AND FUTURE DEVELOPMENTS

Significant progress has been made with the new Delta Tau Geobrick controller on the final Phase II beamlines. On most beamlines we have deployed PMAC co-ordinate systems to drive combined motion systems such as mirrors and slits. We have found this to be very reliable and to provide true synchronised motion at the controller level.

Further work is planned on the EPICS PMAC driver to provide support for trajectory scanning. Eventually we are aiming to have a consistent trajectory scanning interface that is controller- and driver-independent, similar to the existing abstract interface that the motor record provides for standard axis motion.

REFERENCES

[1] http://www.aps.anl.gov/epics/
[2] http://www.opengda.org/
[3] http://www.aps.anl.gov/epics/modules/soft/asyn/
[4] http://www.aps.anl.gov/bcda/synApps/motor/
[5] N.P. Rees, P. N. Denison, T. M. Cobb, "Development of Photon Beamline and Motion Control Software at Diamond Light Source", ICALEPCS 2007, Knoxville, Tennessee, USA.
[6] http://epics-jca.sourceforge.net/caj/
[7] http://ics-web.sns.ornl.gov/edm/
[8] http://www.gmca.anl.gov/TPMAC2/index.html

THE SOFTWARE AND HARDWARE ARCHITECTURAL DESIGN OF THE VESSEL THERMAL MAP REAL-TIME SYSTEM IN JET

D. Alves, A. Neto, D.F. Valcárcel
Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear Laboratório Associado,
Instituto Superior Técnico, Universidade Técnica de Lisboa, 1049-001, Lisboa, Portugal
S. Jachmich
Laboratory for Plasma Physics, Ecole Royale Militaire/Koninklijke Militaire School,
EURATOM-Association "Belgian State", Brussels, Belgium,
Partner in the Trilateral Euregio Cluster (TEC)
G. Arnoux, P. Card, S. Devaux, R. Felton, A. Goodyear, D. Kinna, P. Lomas, P. McCullen,
A. Stephen, K.-D. Zastrow and JET EFDA contributors*
EURATOM-CCFE Fusion Association, Culham Science Centre,
Abingdon OX14 3DB, United Kingdom

Abstract

The installation of ITER-relevant materials for the Plasma Facing Components (PFCs) in the Joint European Torus (JET) is expected to have a strong impact on the operation and protection of the experiment. In particular, the use of all-beryllium tiles, which deteriorate at a substantially lower temperature than the formerly installed Carbon Fibre Composite (CFC) tiles, imposes strict thermal restrictions on the PFCs during operation. Prompt and precise responses are therefore required whenever anomalous temperatures are detected.

The new Vessel Thermal Map (VTM) real-time application collects the temperature measurements provided by dedicated pyrometers and Infra-Red (IR) cameras, groups them according to spatial location and probable offending heat source and raises alarms that will trigger appropriate protective responses. In the context of JETs global scheme for the protection of the new wall, the system is required to run on a 10 millisecond cycle communicating with other systems through the Real-Time Data Network (RTDN).

In order to meet these requirements a Commercial Off-The-Shelf (COTS) solution has been adopted based on standard x86 multi-core technology, Linux and the Multi-threaded Application Real-Time executor (MARTe) software framework. This paper presents an overview of the system with particular technical focus on the configuration of its real-time capability and the benefits of the modular development approach and advanced tools provided by the MARTe framework.

INTRODUCTION

As the fusion scientific community steers their efforts towards the operation of the ITER tokamak, JET aims to provide an important contribution as it will operate with similar PFC materials. In particular, demonstrating the predicted reduction of the tritium retention levels when compared with the previous CFC-based wall[1]. JET's new all metal wall surface made of solid beryllium, tungsten and tungsten coated CFC is much less robust than the previous one therefore raising serious challenges in high power operational conditions.

The Protection of the ITER-like Wall (PIW) project was launched with the aim of providing the necessary tools to ensure the integrity of the vessel during JET's scientific campaigns. These include a set of 13 IR cameras and 9 pyrometer diagnostics that, together with their real-time image processing systems[2], provide the temperatures of PFCs.

The VTM collects these temperature measurements, groups them according to spatial location and probable offending heat source, and raises alarms that trigger the appropriate protective responses coordinated by the Real-Time Protection Sequencer[3] (RTPS) system, see Figure 1. RTPS drives the Local Managers (LM) for all of JET's non-inductive heating systems: Lower Hybrid (LH), Radio Frequency (RF) and Neutral Beam (NB), the Plasma Density Local Manager (PDLM) and the Plasma Position and Current Control (PPCC) system.

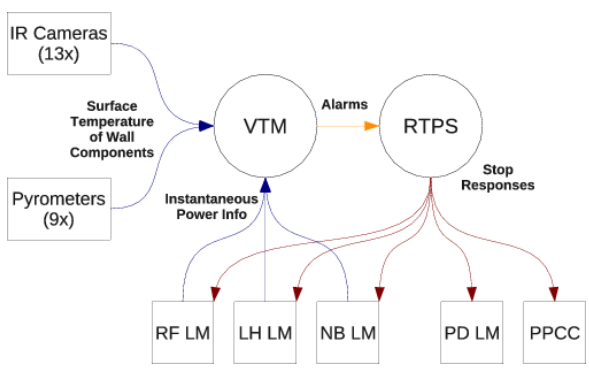


Figure 1: VTM in the context of the PIW project.

* See the Appendix of F. Romanelli et al., Proceedings of the 23rd IAEA Fusion Energy Conference 2010, Daejeon, Korea

VTM FUNCTIONAL OVERVIEW

Temperature measurements of physical tile groups are processed by the VTM in macro sets called Logical Tiles (LTs). These macro sets take advantage of toroidal symmetry to establish the equivalence of temperature measurements that share the same poloidal position although taken at different toroidal locations. This feature allows for different camera views that, although not "looking at" particular physical tiles, monitor "equivalent" ones. Also, in case of measurement loss (e.g. when a camera fails in the middle of an experiment), VTM checks to see if the minimum amount of measurements for each LT is still fulfilled and, if so, the experiment proceeds without significant risk to the machine. After checking the validity of temperature measurements, VTM estimates LT temperatures as the maximum of all temperature measurements within the set.

At a higher level, sets of LTs are grouped into *Wall Segments* (WSs). Similar to the LT estimated temperature, the WS temperature is defined as the maximum of all LT temperatures within the set. Alarm triggering temperature thresholds are placed on WS estimated temperatures. VTM has the capability of processing a maximum number of 60 WSs (30 fixed and 30 user-definable) and more than 400 tiles.

Figure 3 shows part VTM's expert graphical user interface where WS related quantities and the mapping between physical and logical tiles are controlled.

The system was specified to run at 100 Hz, twice the frequency of the real-time image processing units' temperature outputs and its real-time I/O was specified to be performed solely via the ATM-based RTDN[4].

HARDWARE

Because of its technical specifications, and after considering other options such as VxWorks® and VME (VV) or RTAI and ATCA (RA), VTM was chosen to be implemented on a standard 4GB x86 multicore (6 cores) Linux-based PC with a PCI Gb ethernet Network Interface Card (NIC) and a PCI ATM NIC. Despite JET's long history of implementing the VV combination for real-time systems and the recent success of using the RA combination for the new Vertical Stabilisation system, the chosen path proved advantageous in terms of cost, development effort, debugging tools availability and with support within the Linux and MARTe's community.

SOFTWARE

The VTM system was built using the MARTe real-time framework[5] on a multicore vanilla (2.6.35.9) Linux platform. Real-Time performance with a traditionally non real-time operating system is achieved by configuring cpu isolation in terms of processes, threads and Interrupt ReQuest (IRQ) affinities, see Table 1.

The synchronization of the VTM system with JET's central timing is done via the RTDN. Central timing is read by the Real-Time General Services (RTGS) system directly from a VME electronics module and made available to the network. VTM not only time stamps its internal data using this clock but also triggers its own "control" cycle on the arrival of this ATM packet. Preliminary results show that worst case jitters of less than 50 μs on RTGS are propagated to worse case jitters of less than 100 μs on the VTM, see Figure 2.

Table 1: CPU Affinities

CPU #	Task
0	Linux & MARTe services
1	ATM IRQs
2	ATM synch Rx thread
3	ATM Rx threads
4	ATM Rx threads
5	Real-Time thread

Lab tests were performed using the available Linux kernel's real-time patch but those showed no performance improvements when compared to the above mentioned configuration.

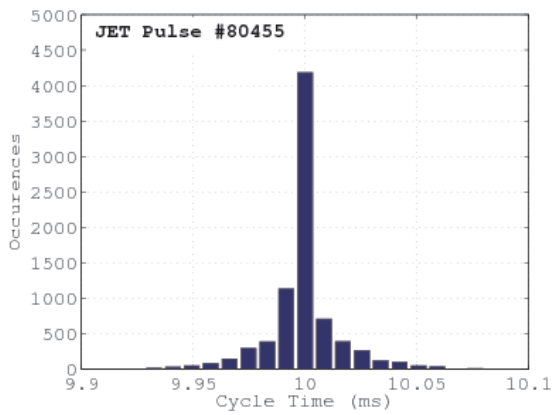


Figure 2: VTM Cycle Time.

Furthermore, the VTM system receives a total of 19 ATM packets in real-time (13 packets from cameras, 2 from pyrometers, 3 from additional heating systems plus the synchronization packet). A standard Linux (kernel module) ATM low-level driver is used together with a (user space) MARTe thread-based high-level driver socket implementation for receiving.

The system has been configured to boot remotely using the *tftp* protocol and mount its file system using NFS. These approaches attempt to make the VTM more resilient to hardware faults.

MARTe is a multi-platform framework for the development and deployment of data-driven, flexible and modular real-time applications. It is based on a real-time oriented C++ library called BaseLib2. It currently supports the VxWorks®, Linux, Linux/RTAI®, Solaris® and Windows® operating systems. Ideally, and assuming the

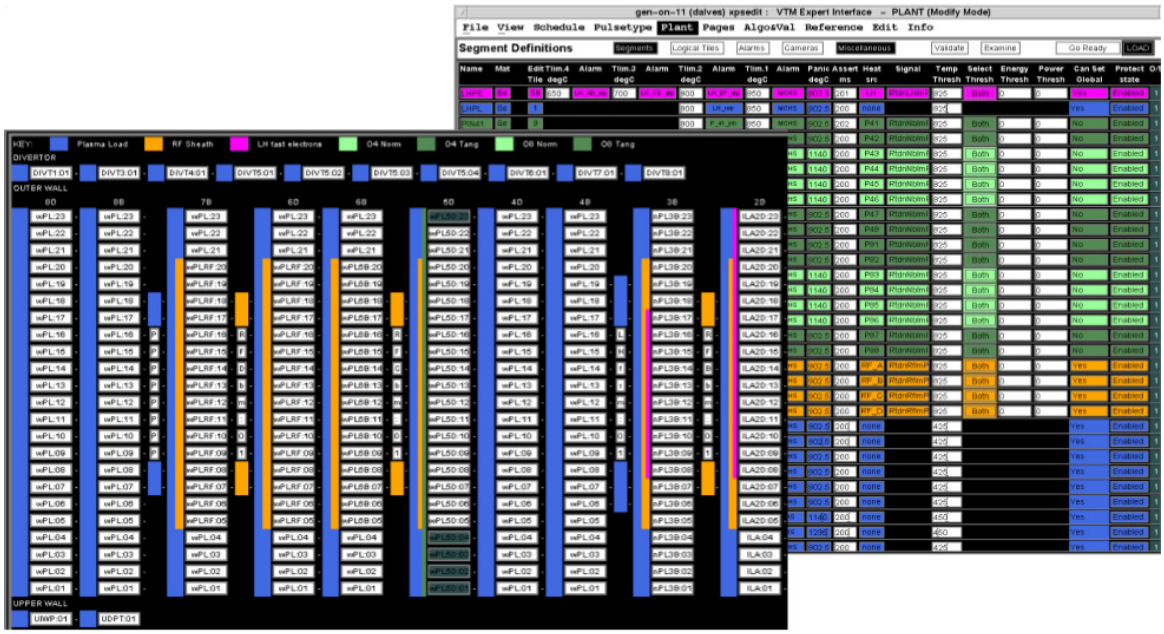


Figure 3: VTM Expert Graphical User Interface.

required hardware components are supported, the developer is only required to write the pieces of software dealing with the logic/algorithmic aspects specific to the control application itself.

MARTe, from the user point of view, is primarily a sequential executor of *Generic Application Modules* (GAMs) in a real-time priority context. MARTe also defines standard high-level interfaces for various activities including driver I/O, http display and messaging. It provides a configurable internal state machine which, in the case of the VTM, is driven remotely by JET's global state machine. According to its current state, the appropriate list of GAMs is executed. Deeply embedded in MARTe's philosophy is the idea that GAMs should not be aware of one another and would perform specific and self-contained tasks communicating with each other exclusively by writing signals to and reading signals from the *Dynamic Data Buffer* (DDB). This buffer is basically a signal pool with a standard access interface. MARTe also contains a driver pool of high-level interfaces for common (and/or maybe less common user implementable) activities, e.g., I/O. Finally, the External Time Triggering Service (ETTS) is the entity that unleashes the execution of the GAMs. It can either be interrupt driven or polling a specific event. Figure 4 shows not only some of MARTe's architecture but also the specific VTM GAMs:

ATM Synch input GAM - is unblocked by the ETTS on the arrival of the ATM synch packet every 10 ms;

ATM input GAMs - make the ATM packets' data available as DDB signals;

Digital Filter GAM - filters DDB signals;

Logical Tile GAM - estimates LT temperatures based on the measurements;

Wall Segment GAM - estimates WS temperatures based on LT temperatures;

Alarm GAM - issues alarms based on WS temperatures and thresholds;

Event GAM - records and displays alarm events;

ATM output GAMs - output the alarms and WS temperatures to the RTDN;

Data Collection GAM - record DDB signals to be collected for posterity;

Web Statistics GAM - http display of DDB signal statistics.

To provide a convenient and intuitive real-time display of the vessel temperature map, an Ajax based http service (see Figure 5) has been developed. Planar images of both JET's inner and outer wall are used as canvas for painting each LTs' temperature using the color code indicated at the bottom of the Figure. This service has been tested with a single browser client polling at 4 Hz and no impact was observed in the system's real-time performance.

PRELIMINARY RESULTS

As a proof of concept, dedicated plasma pulses were performed to test the end-to-end response of PIW's protection apparatus. Simulated camera measurements based on visible light were provided to the VTM and reference pulses were used to place VTM's alarm triggering temperature thresholds at convenient levels. Figure 6 illustrates the achievement of pulse 80455 where a Divertor Hot-Spot (DHS) was detected by the VTM at the bottom of the machine and the correspondent alarm triggered the appropriate predefined PPCC response moving the plasma current's

centroid upwards and reducing the gap between the plasma boundary and the top of the vessel.

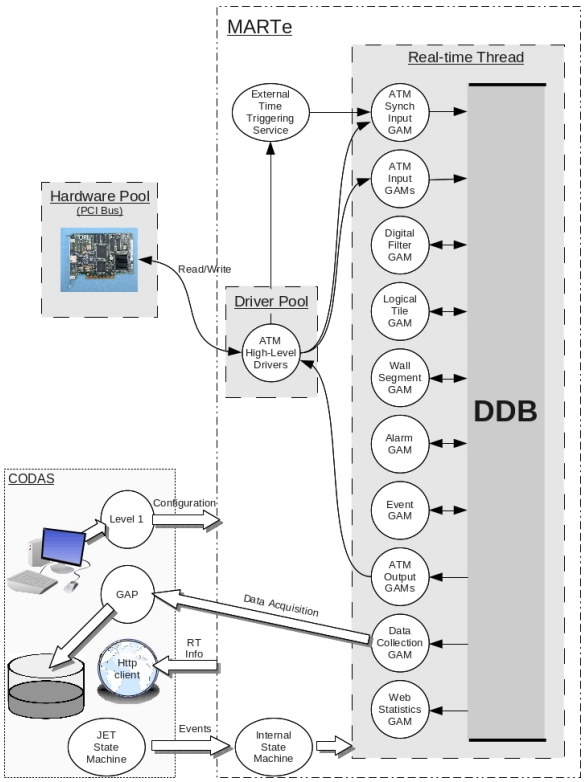


Figure 4: VTM MARTe.

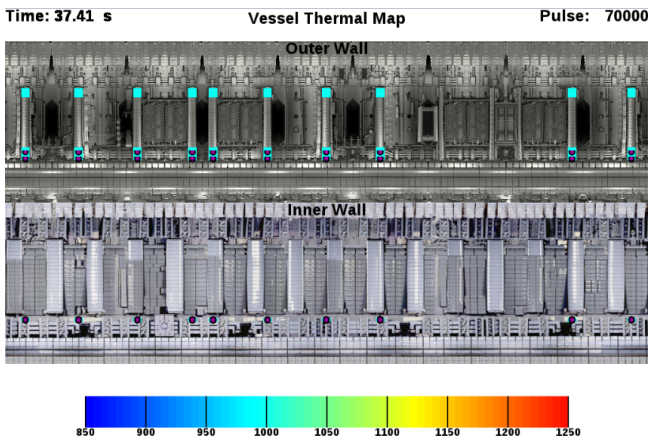


Figure 5: Real-Time temperature map display - http/ajax based polling @ 4 Hz without compromising real-time performance.

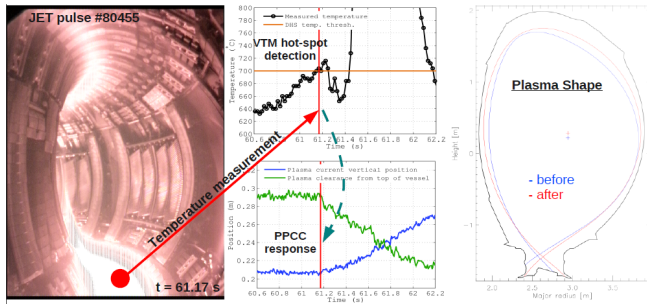


Figure 6: Preliminary results - upon the detection of a thermal event, VTM signaled RTPS, which in turn drove PPCC into a new plasma shape and position.

CONCLUSIONS

The VTM has been demonstrated to perform the role it was designed for in the context of the PIW project. The non traditional configuration of its real-time capability, using a plain Linux vanilla kernel together with the user-space MARTe-based implementation makes it an interesting proof that with little effort it is possible to have an application meeting real-time requirements. Its Ajax-based visual temperature monitoring service with no impact in real-time performance takes this matter to an even higher level demonstrating the potential of the multicore approach.

In the near future this system will be part of the standard machine protection ensemble fundamental to JET operations.

REFERENCES

[1] G. Matthews et al, "JET ITER-like Wall - Overview and experimental programme", Proceedings of the 13th International Workshop on Plasma-Facing Materials and Components for Fusion Applications, Rosenheim, Germany.

[2] M. Jouve et al, "Real-time protection of the ITER-like Wall at JET", this conference.

[3] A. V. Stephen et al, "Centralised Coordinated Control To Protect The JET ITER-like Wall", this conference.

[4] R. Felton et al, "Real-time plasma control at JET using an ATM network", 11th IEEE NPSS Real-Time Conference, Santa Fé 1999.

[5] A. Neto et al, Nuclear Science, IEEE Transactions on 57 (2010).

TIMING SYSTEM SOLUTION FOR MEDAUSTRON; REAL-TIME EVENT AND DATA DISTRIBUTION NETWORK

R. Štefanič, R. Tavčar, J. Dedič, Cosylab, Ljubljana, Slovenia
J. Gutleber, CERN, Geneva, Switzerland
R. Moser, EBG MedAustron, Wr. Neustadt, Austri

Abstract

MedAustron is an ion beam research and therapy centre under construction in Wiener Neustadt, Austria. The facility features a synchrotron particle accelerator for light ions. The timing system for this class of accelerators has been developed in close collaboration between MedAustron and Cosylab.

Mitigating economical and technological risks, we have chosen a proven, widely used Micro Research Finland (MRF) timing equipment and redesigned its FPGA firmware, extending its high-logic services above transport layer, as required by machine specifics. We obtained a generic real-time broadcast network for coordinating actions of a compact, pulse-to-pulse modulation based particle accelerator. High-level services include support for virtual accelerators and a rich selection of event response mechanisms. The system uses a combination of a real-time link for downstream events and a non-real-time link for upstream messaging and non time-critical communication. It comes with National Instruments LabVIEW-based software support, ready to be integrated into PXIe based front-end controllers.

This article explains the high level logic services provided by the real-time link, describes the non-real-time interfaces and presents the software configuration mechanisms.

ABOUT MEDAUSTRON

MedAustron [1, 2] is intended for research and clinical therapy applications. Its synchrotron-based accelerator will have 3-5 ion sources (protons, carbon ions and possibly other light ions) and 5 beamlines, one of which is a rotating gantry. It will provide ion beams with energies up to 800MeV to several irradiation stations used for different purposes. Development follows defined processes and quality management standards. Among the development goals is to use commercially available components where possible to minimize the technological and economical risks. The machine will serve as a blueprint for further applications. Accelerator layout is shown in Figure 1.

TECHNOLOGY CHOICE FOR MEDAUSTRON TIMING SYSTEM HW

No out-of-the box timing system solution exists yet for MedAustron's type of accelerators. The key requirements for the timing system are a deterministic network protocol for real-time operation, reliable event/data distribution and fast response times from the timing system master to all (about 300) controlled devices. It must provide 1µs real-time control loop resolution, 100 ns timestamp resolution and support for 250.000 different accelerator cycles.

When approaching timing system design, we examined the possibilities of reusing existing technology and Micro Research Finland (MRF) [3] proved the most suitable. It is designed for more demanding light-source accelerators and satisfies MedAustron synchronization requirements well. What it lacks is the required higher-level logic and a generic accelerator timing system application. We therefore decided to keep MRF's widely used and proven timing transport layer and build the high level logic application on top in its FPGA firmware. As MedAustron Control System (MACS) software will run on National Instruments' PXIe controllers, we will develop MRF cards in PXIe form factor.

An important part of the timing system is the software support which is fully integrated into MedAustron Control System (MACS). The software provides flexible mechanisms for configuration, control and supervision of the timing system. It is implemented within MedAustron's Front End Control System (FECOS) software framework, developed in LabVIEW, also in close collaboration between MedAustron and Cosylab. The framework is compatible with LabVIEW Real-Time operating system, allowing implementation of both non-real-time and real-time device support applications (i.e. FECOS components).

Reusing the MRF timing transport layer for timing system design enabled us to deal with machine specifics and focus more on integration aspects of the design, which is often an underestimated and overlooked part of timing system development.

TIMING SYSTEM OVERVIEW

MedAustron Timing System, called the Real-time Event Distribution Network (REDNET) comprises a real-time and a non-real-time link. The real-time link is based on MRF transport layer and custom logic in FPGA, whereas the non-real-time link is based on Gigabit Ethernet.

The real-time network topology is the same as original MRF: one Event Generator (EVG) and multiple Event Receivers (EVR), connected in tree topology with multiple fanout layers. Each card is located in a PXI crate

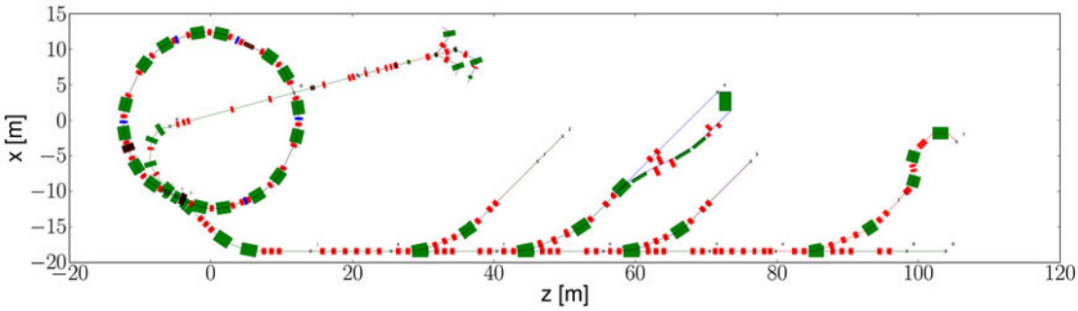


Figure 1: Accelerator layout with injection chain, extraction line and transport lines.

with the PXI controller running support software. The combination of an EVG (or EVR) card with its support software is called the Main Timing Generator (MTG); or Main Timing Receiver (MTR), respectively. The timing system will control about 300 devices, but EVR's sophisticated I/O mechanisms lower the number of required EVRs. Support software provides access to configuration and control to the supervisory control system and triggers transmission of traffic over both the real-time and the non-real-time interface.

The configuration of MTG and MTRs is done by the supervisory control system via a non real-time supervisory interface.

HIGH LEVEL SERVICES OF TIMING SYSTEM REAL-TIME INTERFACE

The following subsections describe the timing system's main services. Richness of logic in communication and event response mechanisms also illustrates specific requirements of MedAustron accelerator class.

Event Response Mechanisms

The timing receiver offers multiple possibilities of controlling its neighbouring cards via numerous hardware connections and via software IRQs. Through LabVIEW API, the user can easily configure the Event Receiver to:

- trigger external devices (via TTL, fibre optic, etc.)
- trigger PXI/PXIe neighbouring cards (via star trigger bus)
- provide neighbouring cards with real-time data (via trigger bus)
- provide SW applications with IRQs and real-time data
- provide delayed, inverted or otherwise "post-shaped" response to timing events (e.g. for fine tuning of synchrotron injection synchronization down to 5 ns resolution)

All types of subscriptions to different events can be used by the same application.

Virtual Accelerator (VAcc) Support

Timing events are scheduled in 5 separate, concurrently usable execution slots (ES), providing concurrently running virtual accelerator functionality. The concept enables users to program individual accelerators as if they were separate, each of them having its own execution configuration. However, timing events of all VAccs are piped through the same serial link (fibre optic), where prioritization takes place in case events are scheduled for emission at the very same moment. Receivers can be linked to any combination of these VAccs and event responses for different ES can be configured separately.

This feature will enable commissioning of different parts of accelerator (e.g. ion sources) at the same time independently, reducing commissioning time immensely. During machine use, it will provide a safe, controllable way of experimenting with new accelerator settings.

Commands

Commands are an extended concept of timing events, also providing data payload, which makes them very useful for distributing near-real-time information. They are implemented using the concept of pipes, additionally abstracting the communication channel to make it even easier to use. Commands have the lowest priority.

Heartbeat Event and Time Service

Along with emission of timing events defined in a sequence, the MTG also provides a unique heartbeat event emitted independently of any other traffic. Its emission range is configurable, ranging from 0.5Hz to 50Hz. Receiver response to the heartbeat event is fully configurable. This functionality is used to operate the injector in temperature stabilized mode.

The time service mechanism provides a transparent way of distributing current time to all receivers.

Real-Time Traffic Prioritization Scheme

Figure 2 explains the prioritization of different real-time traffic and offers an insight into the mechanisms implemented inside EVG FPGA firmware. All emission mechanisms are conveniently configurable through LabVIEW API. The figure shows that among the 5 ES, one has top priority over all other traffic. Next is the heartbeat event, then the other 4 ES, followed by asynchronous user events and commands.

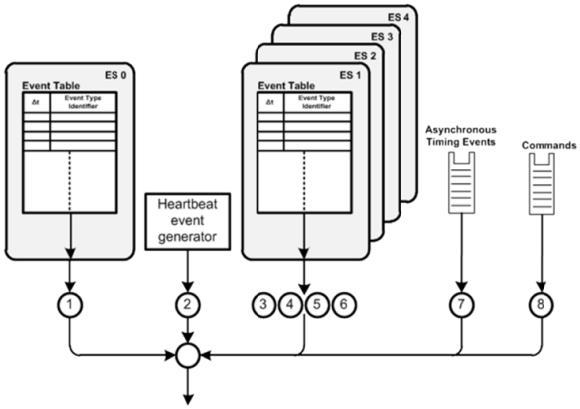


Figure 2: Message prioritization scheme.

Synchronization Parameters

Table 1 shows the timing characteristics of the event emission mechanism.

Timing events can be scheduled with the granularity of 1µs, meaning that every µs, events from all 5 execution slots and the heartbeat event can be scheduled for emission. In such case, fixed prioritization occurs, as shown in Figure 2. At reception, events are time-stamped with 100 ns resolution. These timing characteristics satisfy MedAustron timing system requirements, which strongly guided the system design. The underlying MRF transport layer greatly exceeds this synchronization performance.

Table 1: Timing System Synchronization Parameters

Event Emission Granularity	1µs
Timestamp Resolution	100 ns

NON-REAL-TIME INTERFACE AND SOFTWARE CONFIGURATION

There are two main messaging services implemented for the non-real-time interface. Both are based on National Instruments Simple TCP/IP Messaging (STM), extending its functionality to serve our specific needs.

Simple Messaging Mechanism (SIM)

This is a simple enhancement of STM. It provides the interface to the PVSS-based supervisory control system, which uses it for execution control and configuration of accelerator cycles by issuing requests to the MTG, moving it through the cycle generation process.

Publish- Subscribe Service

Built upon the SIM interface, the Publish-Subscribe Service provides enriched messaging mechanisms for communication between all software support components across the non-real-time network. MTG publishes commands via this interface to all "slow" devices without requirements for deterministic operation. One of the mechanisms based on this service is acknowledgement of commands; for monitoring the system integrity during operation, all commands received by relevant devices via real-time and/or non-real-time interface must be acknowledged via the Publish-Subscribe interface. The MTG denies service if any acknowledgements fail to arrive. The Publish-Subscribe service can freely be used for non-real-time communication between device-specific software components within FECOS.

Configuration of Software Components

Configuration of all FECOS components is provided in XML files, providing startup and operation parameters, process variable locations and any other user-defined information. In fact, all structurally rich information is distributed to and among components in standard XML format. This information includes accelerator sequence definitions used by the MTG and timing event response configurations used by the EVR.

Process variables for online configuration and monitoring of end devices are implemented via LabVIEW's shared variable engine and connected via OLE for Process Control (OPC) to the PVSS-based SCADA server of the supervisory interface.

CONCLUSION

By using the off-the-shelf product from MRF and building the machine specific real-time application on top of its already widely used and stable transmission layer, the overall design time can be greatly reduced. The design effort can thus be strongly focused on specifics of the machine itself and not on the timing distribution layer, which has already been successfully addressed multiple times before. The total time from requirements to the working system was less than 2 years. This design approach is the way to achieve precise, but versatile solutions able to cope with complex use cases of different machines.

The control system software framework provides developers with flexible communication and configuration mechanisms. It also provides a generic component structure which readily integrates into the control system framework and allows the device support developer to focus mainly on the implementation of device-specific application, speeding up the development process.

REFERENCES

[1] M. Benedikt, A. Wrulich, MedAustron—Project overview and status, Eur. Phys. J. Plus (2011) 126: 69

[2] M. Benedikt, A. Fabich, "MedAustron — Austrian hadron therapy centre", Nuclear Science Symposium Conference Record 2008. NSS '08. IEEE, pp.5597-5599, 19-25 Oct. 2008

[3] J. Pietarinen, "MRF Timing System", Timing Workshop CERN, February 2008

SYNCHRONOUSLY DRIVEN POWER CONVERTER CONTROLLER SOLUTION FOR MEDAUSTRON

Luka Šepetavc, Jože Dedič, Rok Tavčar, Cosylab, Ljubljana, Slovenia
Johannes Gutleber, CERN, Geneva, Switzerland
Roland Moser, EBG MedAustron, Wr. Neustadt, Austria

Abstract

MedAustron is an ion beam cancer therapy and research centre currently under construction in Wiener Neustadt, Austria. This facility features a synchrotron particle accelerator for light ions. Cosylab is closely working together with MedAustron on the development of a power converter controller (PCC) for the 260 deployed power converters – power supplies.

Power converters deliver power to magnets used for focusing and steering particle beams. We have designed and developed software and hardware which allows integration of different types of power converters into MedAustron's control system (MACS). PCC's role is to synchronously control and monitor connected power converters. Custom real-time fibre optics link and modular front end devices have been designed for this purpose. Modular front end devices make it possible to interface with almost any type of power converter – with or without built in regulation logic. We implemented real-time mechanisms and a dedicated real-time fibre link to satisfy requirements for synchronous control of power converters and data acquisition of their output current measurements.

MEDAUSTRON

MedAustron [1, 2, 3] is a synchrotron based accelerator for cancer treatment with protons and carbon ions. It will be used for clinical and non-clinical research in the fields of medical radiation physics, radiation biology and experimental physics. Synchrotron will deliver proton beams with energies up to 800 MeV. Three irradiation rooms are indented for medical use, one of which with a gantry, and one irradiation room for non-medical use.

INTRODUCTION

PCC is a distributed system of PXIe crates and front end devices that controls power converters. Its purpose is to ensure precise application of desired output values on power converters at precise points in time. Output value on power converters determines the magnetic field in magnets throughout the accelerator. PCC works together with MedAustron's timing system to achieve synchronous operation across all power converters. An important part of PCC is acquisition of measurements and verification of measured values against desired reference values.

POWER CONVERTER CONTROLLER – SYSTEM OVERVIEW

Output current values on power converters are set via set-points. PCC's main role is synchronous generation of set-points and acquisition of measurement results. During normal accelerator operation PCC works autonomously. Its operation is mainly dictated by the timing system. For expert or service use, PCC allows completely manual control of its functions. All of PCC's configuration parameters can be changed during runtime, which greatly simplifies and shortens commissioning and servicing of the machine.

Slow Power Converters

Slower power converters, such as those for magnets in low energy beam transport (LEBT), medium energy beam transport (MEBT) and extraction line, are controlled with single set-point values. Output on these power converters usually changes every few seconds or every few minutes. This change of output value coincides with accelerator cycles. Every accelerator cycle defines a different desired energy level of the beam.

Single set-point values are provided to the PCC in advance as a list of values associated to accelerator cycles. Optionally, when PCC is operated manually, single set-point values can be provided directly over the operator's graphical user interface and can be applied on power converter's output immediately.

PCC can provide single set-point values in different units. For example when the front end device is coupled with a digital signal processor (DSP) board, set-points can be provided as voltage, current or magnetic field. This in-house developed DSP board internally processes the received value and puts an appropriate voltage level on its analog output interface towards the power converter.

Fast Power Converters

Faster power converters, used for scanning magnets and in the main ring, are controlled with set-point sequences. Set-point sequence is a set of multiple single set-point values stored in one file. The execution of a sequence is dictated by the timing system. Graphical representation of a sequence is shown in Fig. 1.

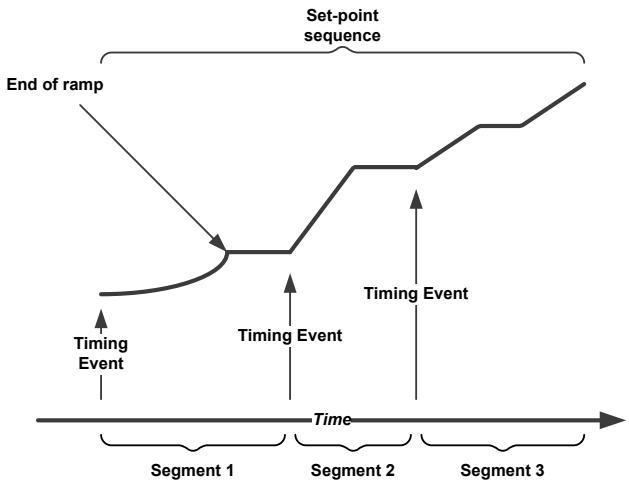


Figure 1: Structure of a set-point sequence used for fast power converters.

Output Waveform Verification

In order to verify the output of power converters against the desired reference values, PCC acquires measurements from connected power converters. Measurements are internally buffered in the PCC and optionally filtered. PCC can perform automatic verification of measured values against reference values or provide measurements to the user for inspection. Acquisition of measurements can be triggered with timing events, thus acquiring results from a specific point in time. For faster power converters, measurements are acquiread and time stamped with an accuracy of 1 microsecond.

Measurements are updated and visualized to the user in real-time, but can also be frozen. Results can be overlaid with a reference waveform from the generated set-point sequence and received timing events. This representation contains a complete set of data for the user to analyse and take advantage of in the process of fine tuning machine parameters.

Flexible Support for New Power Converter Types

Power converters with built-in logic and known set of commands can be easily integrated into the PCC. Minimal effort is required to implement a new driver in PCC application. The role of the driver is to translate power converter's set of commands into a format that is understood by the PCC application. Driver handles power converter's low level functionality and specific initialization procedures. Higher level functionality is handled by the PCC application above the driver. For expert usage and service purposes, PCC can bypass power converter driver and transparently pass commands to the power converter in raw format.

Power converters, which act as regular voltage sources, are integrated into the PCC seamlessly. Such power converters are connected to the DSP board which is attached on the front end device. Driver for the DSP board already exists and allows full configuration of parameters, e.g. for optimization of regulation loop.

TECHNOLOGY BEHIND THE POWER CONVERTER CONTROLLER

To successfully operate power converters in MedAustron's synchrotron, several challenges needed to be overcome. The most important one is synchronicity of PCC's actions. A group of fast power converters has to be synchronously controlled with an accuracy of 1 microsecond or better. Different synchronization mechanisms have to be supported. The main synchronization is dictated by the timing system, from which PCC receives timing events. Other types of synchronization include pulse synchronization and start/stop synchronization. They can be used in combination with respiratory system which is monitoring the patient and can, for example, pause and continue the operation of the PCC.

PCC Application Software

Along with hardware we designed and developed full software support. The main part of the PCC system is based on an off-the-shelf National Instruments PXIe crate, which can be seen in the middle of Fig. 2.

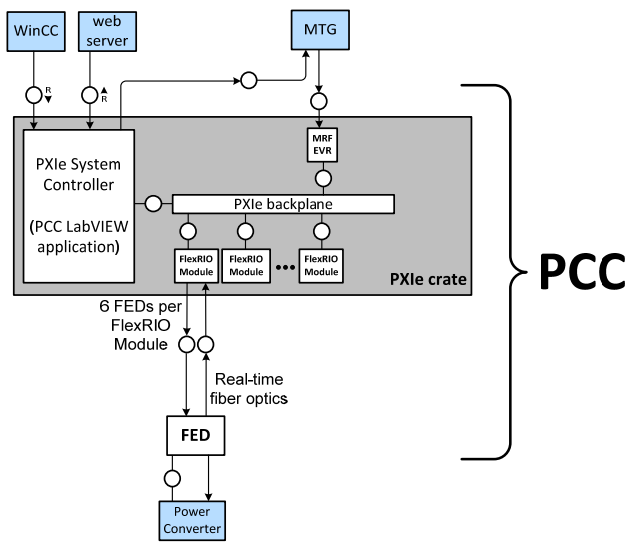


Figure 2: Architecure of PCC and interaction with other systems.

Running on the PXIe system controller are multiple instances of PCC application, integrated into MedAustron's Front End Control System (FECOS) software framework, all developed in LabVIEW. In MedAustron, PCC application is controlled and supervised over SIMATIC WinCC OA SCADA system. For expert use, PCC application can optionally be controlled with a web interface. PCC application is responsible for handling data from supervisory control system, controlled hardware and timing system, as well as specifics of different power converters: translation of

power converter commands, initialization procedures, preparation of set-point values etc. For every power converter there is a dedicated PCC application instance running on the system controller. Every PCC application instance is configured with a configuration XML and configuration shared variables accessible over the network. When more power converters need to be connected, additional PCC applications are instantiated. If all slots in the PXIe crate are full, additional crate(-s) are added.

PCC application communicates with FlexRIO FPGA Modules – FPGA based cards inserted into PXIe crate slots. Firmware on FlexRIO FPGA Modules carries out time critical tasks. These tasks include transmission of data over real-time optical link, reception of timing events from the PXIe timing receiver card, generation of single set-point values and sequences, acquisition and buffering of measurement values etc. Attached onto FlexRIO FPGA Modules are FlexRIO Adapter Modules carrying optical interfaces for communication with front end devices.

FlexRIO Adapter Module with Generic Optical Interfaces

FlexRIO Adapter Module (Fig. 3) is a custom developed adapter board with 6 generic optical connectors that connects up to 6 front end devices. The adapter is completely generic and can be used for any type of application that requires multiple optical interfaces.

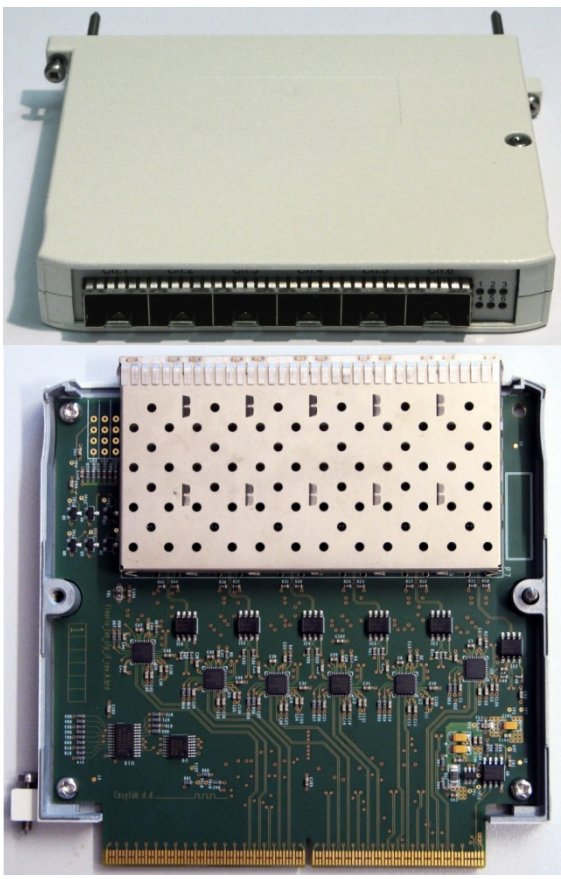


Figure 3: FlexRIO Adapter Module with 6 generic optical connectors (top – encased, bottom – PCB).

FlexRIO Adapter Module attaches onto a FlexRIO FPGA Module which takes control of the optical interfaces.

Front End Device

Front end device (Fig. 4) is an FPGA-based board which connects to a power converter directly through a RS-422 serial interface or indirectly through a DSP board. Front end device acts as a translator between the real-time fibre link and power converter's interface.

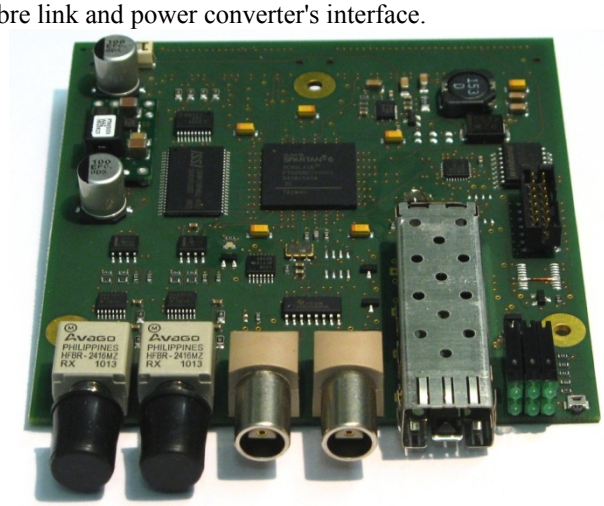


Figure 4: FPGA based front end device board.

FED plugs onto an in-house developed DSP board in case a regular voltage source power converter needs to be controlled. DSP board implements sophisticated regulation logic and regulates the output of the power converter via an analog interface.

Front end device can be located a few hundred meters away from the PXIe crate, near the actual power converter. It plugs onto a baseboard which acts as an expansion board for different I/O connectors (RS422, UHPI, trigger input and output...). Because most of the complex logic is located on the front end device, it makes it easier to design a new baseboard with connectors, if new types of interfaces are required.

Real-Time Fibre Optic Link

A custom real-time fibre optic link protocol has been designed for communication between FlexRIO Adapter Modules and front end devices. Its generic design allows it to be used for different other purposes. Its main feature is logic pipes. Each pipe has an independent interface for transmission and reception of data. Data is prioritized among the pipes by the protocol. One of the pipes has the highest priority and provides completely deterministic performance in terms of latency when data is transmitted and received. This pipe is used for transmission of set-point values (current, voltage...) towards power converters and reception of measurement values from power converters.

All logic pipes and the protocol are designed in a generic way. The protocol can be easily reused for other

application requiring deterministic behaviour, plus non-deterministic transmission of background data.

CONCLUSION

Power converter controller for MedAustron is a distributed system based on off-the-shelf National Instruments PXIe hardware and custom designed modular hardware components. Modular front end device and baseboard design simplify integration with different types of power converters. Adding software support for new types of power converters requires minimal effort. PCC integrates into WinCC SCADA system and is completely configurable during runtime. PCC's operation is dictated by the timing system. Custom designed real-time fibre link protocol guarantees 1 microsecond synchronisation among different power converters. The PCC distributed system allows control of an arbitrary number of power converters of various types, making it suitable for machines of different sizes.

REFERENCES

[1] M. Benedikt, A. Wrulich, MedAustron — Project overview and status, Eur. Phys. J. Plus (2011) 126: 69

[2] M. Benedikt, A. Fabich, MedAustron — Austrian Hadron Therapy Centre, Nuclear Science Symposium Conference Record 2008. NSS '08. IEEE, pp.5597-5599, 19-25 Oct. 2008

[3] M. Benedikt, Status of MedAustron, Symposium Verein AUSTRON, Vienna, March 15, 2011

PCI HARDWARE SUPPORT IN LIA-2 CONTROL SYSTEM

D. Bolkhovityanov, P. Cheblakov, BINP, Novosibirsk, Russia.

Abstract

LIA-2 control system is built on cPCI crates with x86-compatible processor boards running Linux. Slow electronics is connected via CAN-bus, while fast electronics (4MHz and 200MHz fast ADCs and 200MHz timers) are implemented as cPCI/PMC modules. Several ways to drive PCI control electronics in Linux were examined. Finally a userspace drivers approach was chosen. These drivers communicate with hardware via a small kernel module, which provides access to PCI BARs and to interrupt handling. This module was named USPCI (User-Space PCI access). This approach dramatically simplifies creation of drivers, as opposed to kernel drivers, and provides high reliability (because only a tiny and thoroughly-debugged piece of code runs in kernel).

LIA-2 accelerator was successfully commissioned, and the solution chosen has proven adequate and very easy to use. Besides, USPCI turned out to be a handy tool for examination and debugging of PCI devices direct from command-line. In this paper available approaches to work with PCI control hardware in Linux are considered, and USPCI architecture is described.

INTRODUCTION

LIA-2 [1] linear induction accelerator is designed in Budker INP as an injector for full scale 20 MeV linear induction accelerator which can be used for X-ray flash radiography with high space resolution. This machine utilizes ultra high vacuum, precise beam optics design based on low temperature dispenser cathode of 190 mm in diameter. The designed value of beam emittance (120 π mm•mrad, not normalized) is achieved at 2 MeV and 2 kA of electron beam energy and current.

LIA-2 control system is based on cPCI crates with x86-compatible CPU boards running Linux (CentOS-5.2 with custom-built 2.6.25 kernel. Slow control electronics (modulator controllers, slow DACs/ADCs) is connected via CAN, while fast hardware – such as 4MHz and 200MHz ADCs and 200MHz timers – are cPCI/PMC boards.

LIA-2 control system software is based on CX [2]. Being highly modular, CX doesn't include CAN and PCI support "out of the box". So, PCI hardware support had to be created from scratch.

Control system hardware drivers differ dramatically from regular OS drivers in many aspects.

- They implement different model: interaction via "channels" or "records" instead of files, char/block semantics, ioctl()s etc.
- Control system hardware itself, such as DACs, ADCs, timers etc., also has very little in common with regular computer hardware, such as NICs, display cards, serial and SCSI/IDE/SATA ports.
- So, timing requirements and model of operation also differ radically.
- Usually what CS drivers do is read and write hardware memory, plus catch IRQs in a simple manner - what we used to do with CAMAC and VME.

But PCI subsystem in Linux kernel doesn't have a simple way to do it - it is oriented on regular hardware driver requirements.

So, we had to make a way to access PCI hardware in the same manner as CAMAC and VME.

ANALYSED SOLUTIONS

The following existing solutions were considered while choosing the most appropriate one for LIA-2.

Loadable Kernel Modules

It is also known as kernel drivers. A module, running in kernel space, has direct access to PCI and various kernel subsystems. Thus, it can support arbitrary complex hardware, and achieve maximum performance. However, this comes at very high price:

- Development of kernel code is an extremely complex task. Kernel programming requires exceptionally high skills, and debugging such code is very cumbersome.
- Kernel modules are closely tied to internal kernel interfaces, which often change between versions. So, any kernel upgrade requires modification and repeated testing of driver code.
- Similarly, any modifications due to changes in control hardware are also labour-intensive.
- Moving to another *nix platform is almost impossible.

UIO: User-Space Drivers

UIO is a framework that allows implementing PCI drivers in userspace [3]. User space drivers eliminate most of disadvantages of kernel drivers. Advantages of userspace drivers are:

- you don't have specific kernel drivers for wide range of similar devices;
- much easier to develop and to port to other *nix operation systems;
- fault tolerance.

A tiny kernel-side driver to handle some basic interrupt routine is needed as part of every UIO driver. UIO is just a simple way to create very simple, non-performance critical drivers, which has probably been merged more with a "merge-and-see-if-it-happens-something-interesting" attitude than anything else. For now UIO doesn't allow to create anything but very simple drivers: no DMA, no network and block drivers [4].

Microdrivers

The Microdrivers [5, 6] architecture was developed at University of Wisconsin-Madison. The main goal of microdrivers is to achieve a better fault tolerance without loose in performance. Microdrivers reduce the amount of driver code running in the kernel by splitting driver functionality between a small kernel-mode component and a larger user-mode component.

Microdrivers seek the middle ground between monolithic kernels and microkernels, and improve reliability while maximizing performance. In a microdriver, the functionality of a device driver is split between a kernel-mode component and a user-mode component. The kernel-mode component contains critical and frequently used functionality, while the user-mode component contains non-critical and infrequently used functionality.

However, this approach is still destined for "usual" devices, not for control system specific hardware. And yet this requires a kernel part for *each* driver.

Hardware Access Layer

Many major hardware manufacturers implement their own, rather complex, infrastructures to access their control hardware. Usually these include a set of kernel drivers and multi-layer user-space libraries.

Examples include NI-KAL from National Instruments [7], IX-PCI, IX-ISA, IX-PIO from ICP-DAS [8]; COMEDI [9] was an open-source attempt to make manufacturer-agnostic implementation of such system.

Those infrastructures are often just components of larger systems, are usually oriented on specific manufacturer's hardware, and in many cases include closed-source parts.

Thus, this approach looks inadequate for our goals.

WinDriver™

The WinDriver™ [10] product line supports any device, regardless of its silicon vendor, and enables you to focus on your driver's added-value functionality, instead of on the operating system internals.

Its advantages are simple development of drivers and their portability between platforms.

But WinDriver is not free and it is closed-source.

USPCI APPROACH

USPCI (User-Space PCI access) implements approach, similar to WinDriver's: drivers run in user-space, and access hardware via a single tiny kernel module, which implements PCI I/O and simple interrupt management.

The main goal was to minimize drivers' complexity and design time. User-space drivers don't need to implement standard low-level interfaces, as regular OS drivers do. Since most control hardware don't require high throughput and ultra-low, real-time-like interrupt response latency, user-space implementation is quite adequate.

This approach has been widely used in Linux.

- Libusb [11] is a good example: it allows implementing user-space drivers for wide class of USB devices.
- X.org's 2D drivers operate in userspace, thus simplifying support of different unix-like OSes.
- CUPS and programs accessing the serial port like pppd are yet another example of userspace programs accessing the devices directly - the kernel doesn't implement any specific LPT printer or serial modem driver, those userspace programs implement the driver that knows how to talk to the printer.

USPCI INTERNALS

USPCI is a small and simple kernel module, implementing following functionality:

- /dev/uspci char device;
- access to address space of PCI-device (I/O ports & memory);
- data read and write in any PCI BAR of selected device (regardless of type – memory and I/O ports are accessed in the same way);
- 8-, 16-, 32-bit I/O support (64-bit is implemented, but yet untested);
- interrupt handling, with shared IRQ support (PCI address and mask+condition are specified for IRQ checking);
- interrupt counting;
- interrupt flags accumulation;
- target PCI device can be chosen by either PCI address (BUS:DEVICE.FUNCTION) or device ID (VENDOR:DEVICE:INSTANCE or VENDOR:DEVICE:SERIAL).

Since USPCI kernel module is simple and short, it was easy to test and is stable.

USPCI API and Workflow

The USPCI module is accessible from user-space via */dev/uspci* char device. Most operations are performed via *ioctl()* interface.

The following steps should be performed first to interact with PCI device:

- Open the */dev/uspci* via *open()* syscall.
- Select target PCI device – either by bus ID, or by vendor:device ID. This is achieved via USPCI_SETPCIID ioctl.
- Optionally specify a method of IRQ checking (required for shared IRQs) and IRQ acknowledgement. USPCI_SET_IRQ does this.
- Optionally specify a list of operations to perform upon close of file descriptor (such as stopping, IRQ masking, etc.) - USPCI_ON_CLOSE.

Than device's resources can be read and written via USPCI_DO_IO ioctl; very similar to how NAF is done in CAMAC or bus I/O is performed in VME. Usually all these *ioctl()*s are wrapped by a more friendly library.

In order to "finish" interrupt processing the USPCI_FGT_IRQ is called, which returns interrupt count

and accumulated interrupt flags; these are reset afterwards.

USPCI implements “poll” interface, which allows using file descriptor, associated with device, in *select()* and *poll()* syscalls.

To finish interaction with device, its file descriptor must be *close()*'d. Since *nix closes all file descriptors upon program termination, this, together with ON_CLOSE specification, enables to leave device in a safe state in all cases, including user-space driver crash.

Uspci_Test

Command-line interface program was developed for the purpose of device debugging. This program provides an access to all USPCI functions and logic. The use of the program allows analysing of device operation at development of userspace drivers.

CONCLUSION

USPCI Linux kernel driver was developed at BINP for access to PCI-devices of LIA-2 control system. This kernel driver allows developing of userspace drivers that is an optimal approach for control system building.

The drivers for following BINP-made devices were developed with the use of USPCI: timer DL 200 ME Fast, timer DL 200 ME Slow and ADC 200 ME. These drivers are successfully implemented in CX control system for LIA-2. The driver for PISO-ENCODER600 of ICP-DAS Co., Ltd production was also developed [12].

REFERENCES

[1] P. Logachev, A. Akimov, P. Bak et al., “Perfomance of 2 MeV, 2 ka, 200 ns linear induction accelerator with Ultra low beam emittance for X-ray flash radiography”, IPAC'11, Kursaal, San Sebastian, Spain, September 2011, WEoAA02

[2] D. Yu.Bolkhovityanov, A.Yu.Antonov, R.E.Kuskov, “Present Status of VEPP-5 Control System”, PCaPAC2006, Newport News, VA, USA

[3] UIO: user-space drivers, http://lwn.net/Articles/232575/

[4] Linux 2 6 23, http://kernelnewbies.org/Linux_2_6_23

[5] Vinod Ganapathy, Arini Balakrishnan, Michael M. Swift, Somesh Jha, “Microdrivers: A New Architecture for Device Drivers”, HotOS XI, San Diego, California, USA, May 2007

[6] Vinod Ganapathy, Matthew J. Renzelmann, Arini Balakrishnan, Michael M. Swift, Somesh JhaThe, “Design and Implementation of Microdrivers”, ASPLOS XIII, Seattle, WA, USA, March 2008

[7] http://joule.ni.com/nidu/cds/view/p/id/2372/lang/en

[8] http://www.icpdas.com/download/linux.htm

[9] COMEDI, Linux control and measuremnt device interface http://www.comedi.org/

[10] WinDriver™, http://www.jungo.com/

[11] libusb, www.libusb.org/

[12] PISO-ENCODER600, PCI Bus, 6-axis Encoder Input Card, http://www.icpdas.com/

PERFORMANCE OF THE STANDARD FAIR EQUIPMENT CONTROLLER PROTOTYPE

Stefan Rauch, Ralph C. Bär, Wolfgang Panschow, Matthias Thieme, GSI, Darmstadt, Germany

Abstract

For the control system of the new FAIR accelerator facility a standard equipment controller, the Scalable Control Unit (SCU), is presently under development. First prototypes have already been tested in real applications [1]. The controller combines an x86 COM Express™ Board and an Altera Arria™ II FPGA. Over a parallel bus interface called the SCU bus, up to 12 slave boards can be controlled. Communication between CPU and FPGA is done by a PCIe link. We discuss the real time behaviour between the Linux OS and the FPGA Hardware. For the test, a Front-End Software Architecture (FESA) class, running under Linux, communicates with the PCIe bridge in the FPGA. Although we are using PCIe only for single 32 bit wide accesses to the FPGA address space, the performance still seems sufficient. The tests showed an average response time to IRQs of 50 μs with a 1.6 GHz Intel Atom CPU. This includes the context change to the FESA user space application and the reply back to the FPGA. Further topics are the bandwidth of the PCIe link for single/burst transfers and the performance of the SCU bus communication.

DESCRIPTION OF THE SCU

- SCUB connections with LVDS and PCIe
  - strict Master/Slave bus
  - 16Bit data bus, 16Bit address bus
  - in addition serial LVDS and PCIe channels
- COM Express™ module Type II with Intel Atom
- 32Mbyte Parallel Flash
- RJ45 Gigabit Ethernet from COM Express™ module
- two SFP connectors
  - Timing (White Rabbit over GbE)
  - AUX LAN (GbE)
- Arria™ II FPGA
- White Rabbit Receiver (FPGA) [2]
- 128Mbyte DDR3 memory
- IO Extension for existing timing system
  - DEV-BUS: GSI fieldbus
  - Event-In: Timing Events
  - both based on MIL-STD-1553
- USB 2.0
- 2x EIA-232

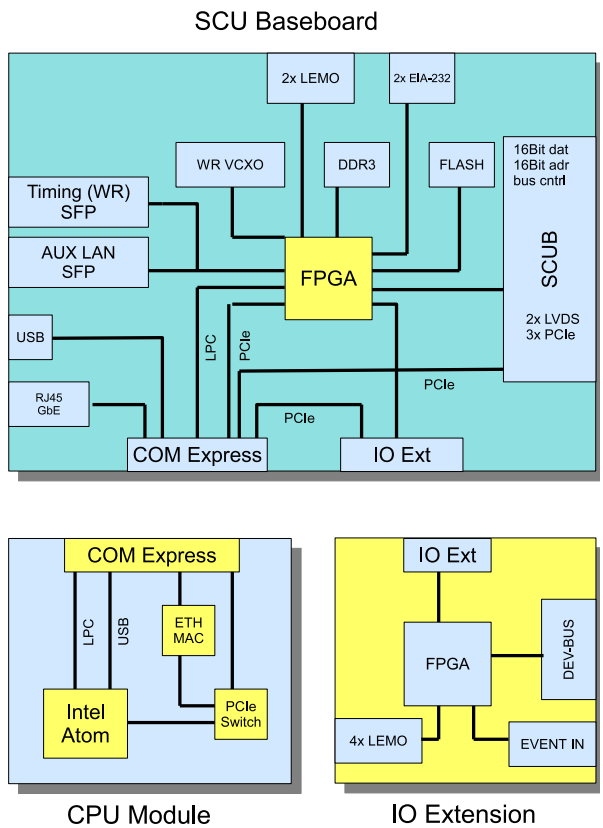


Figure 1: Layout of the SCU Carrier Board.

STATUS REPORT OF THE PROJECT

The prototype should have been ready in July 2011, but to complications during the design phase it came to a delay. The layout for the prototype baseboard is now finished and ready for production (see Figures 1, 2, 3). The first boards are expected to arrive in November 2011.

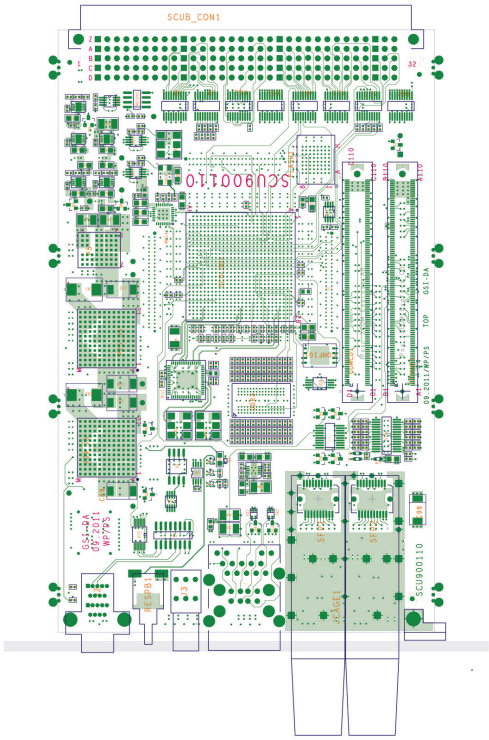


Figure 2: Layout of the SCU Carrier Board.

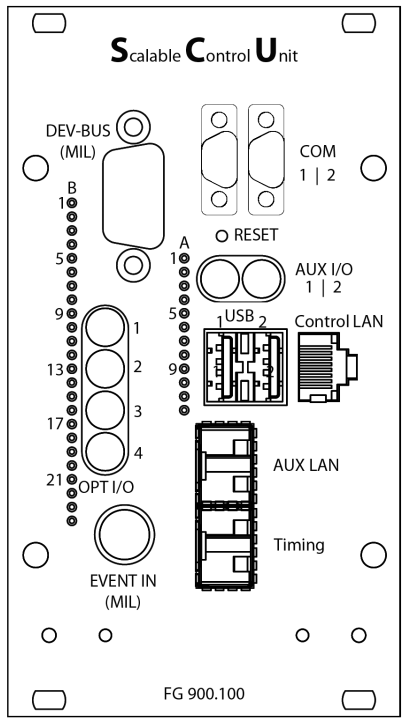


Figure 3: Front Panel of the SCU Cassette.

PROBLEMS DURING DESIGN PHASE

Early during the design phase came the decision for the combination of an FPGA and an COM Express™ module. The connection of the Module requires an FPGA with PCIe capability. The decision fell for the Altera Arria™ II FPGA with high speed serial transceivers. The Arria™ FPGA comes with a hard IP cell for PCIe so no license fees are needed. This FPGA and the design tools were quite new, when the testing of base components of the system like PCIe or DDR3 communication began. We had the possibility to test these technologies in another project and realized some problems with the design tools. In FPGA prototyping you typically implement the design first in the FPGA due to constraints in the pinning of the PCB. In this special case, the design tool allowed the hard IP core to be connected to the transceiver block 1 of the Arria™. The problem is, there is no hardware connection in the FPGA and the PCBs did not run PCIe with the hard macro cell. One possibility to solve this problem was the use of a soft core implementation. This would result in high license fees and unnecessary loss of logic cells in the FPGA. Another temporary solution was the rewiring on the PCB. After patching the PCIe to transceiver block 0 the connection worked perfectly. Altera fixed the issue in the next version of their design tool, but for the final solution a redesign of the PCB was needed.

Another problem worth mentioning was the DDR3 controller. With the first version of the design tool we could not get an reliable system talking to an DDR3 memory. Only after applying several patches the DDR3 accesses were working. But on the PCB we used for testing PCIe, the DDR3 gave us some more trouble. The chip we used is an 16Bit wide DDR3, but only 8Bit wide accesses were successful. This issue could be solved with the newest design software.

LESSONS LEARNED

If we had to do the project again, we would plan more time for testing the new (at least for us) technologies. We were aware of the fact, that their would be problems with a new and complex FPGA like the Arria™ II and this was not the first FPGA project we did. But when dealing with FPGAs like the Arria™ II, there are a lot of constraints in placing the external connections. This is especially true for connecting the pins of the memory controllers. The same applies for the high speed serial transceivers and the clocking of these. You have to be totally sure, how you are going to use use the transceivers, because most changes cannot be done after finishing the PCB. This was a new experience for us as we have only dealt with low scale FPGAs like Altera Cyclon™ or Xilinx Spartan™ before. The vendors are aware of the problem and try to help with wizards for pin placing and constraining. During layout phase this tools can be used again to find optimal routes on the PCB.

IN DETAIL: IMPLEMENTING A LPC UART

- Problem: SuperIO Chip does not fit in Layout
- Solution: extending LPC bus to FPGA, implementing UART functionality in logic
- Low Pin Count Bus, four address/data lines, serial clk, frame signal

The COM Express™ module of the SCU needed an external Winbond SuperIO chip for the UART functionality. This is mainly needed for debugging with a serial console during the boot phase. The BIOS of the module supports up to two UARTs. During the layout phase of the board it came clear, that there is no space left on the base board for the SuperIO chip. This and the fact, that only the UART functionality of the chip is used, lead us to moving the logic into the FPGA. The communication between SuperIO and COM Express™ module uses the LPC bus [3]. This bus is a serialized version of the ISA bus and uses four address/data lines and a serial clock. It runs at 33Mhz and offers bus modes as e.g. IO read/write, MEM read/write. The bus lines are connected to the FPGA instead of the SuperIO. In the FPGA we use logic for decoding the LPC bus accesses to UART transfers. The logic modules are based mainly on existing projects from OpenCore.org: a Whishbone LPC Peripheral bridge [4] and a 16750 compatible UART [5]. If you want to emulate the behaviour of the Winbond SuperIO chip, you have to understand, how the extended addressing of the SuperIO works. The chip is connected to the IO port space of the COM Express™ module. It supports up to 11 logical devices (e.g. UARTs, printer port, MIDI). To access the configuration registers of these devices you have to enter the extended function mode. This is done with two writes of 0x87 to the base address of the SuperIO. Then you select the device number and the configuration register. After reading or updating the register content you send the value 0xAA to the base address. This mechanism has to be implemented in the FPGA to allow the BIOS to configure the logical devices, like setting a base address of the UART. The 8 UART registers of the 16750 compatible core are then mapped to the configured base address. The device can now be accessed by the BIOS code during boot up and by an operating system driver. It is possible to implement further devices in the FPGA like another UART or printer port with this scheme.

REFERENCES

[1] M. Thieme, S. Rauch, M. Zweig, W. Panschow, "Single Board Computer for Equipment Control", ICALEPCS2009, Kobe, October 2009, TUP060, http://www.JACoW.org.

[2] P. Moreira, J. Serrano, T. Włostowski, P. Loschmidt, G. Gaderer, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet",
IEEE Precision Clock Synchronization for Measurement, Control and Communication 2009, pp1-5, Brescia, October 2009

[3] Low Pin Count Interface Specification
http://www.intel.com/design/chipsets/industry/lpc.htm, last visited on 5.10.2011.

[4] Wishbone LPC Peripheral Bridge
http://opencores.org/project,wb_lpc, last visited on 5.10.2011.

[5] 16750 compatible UART
http://opencores.org/project,uart16750, last visited on 5.10.2011.

NEW DEVELOPMENTS ON TORE SUPRA DATA ACQUISITION UNITS

F. Leroux, G. Caulier, L. Ducobu, M. Goniche
CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France
G. Antar, American University of Beirut, Physics Department, Beirut 1107 2020, Lebanon

Abstract

The Tore Supra data acquisition system (DAS) was designed in the early 1980s and has considerably evolved since then. Three generations of data acquisition units still coexist. As cost and maintenance of different operating systems is expensive, it was decided to explore an alternative solution based on an open source operating system (OS) with a diskless system for the fourth generation. In 2010, Linux distributions for VME bus and PCI bus systems have been evaluated and compared to Lynx OS™ real time OS. The results obtained allowed to choose a version of Linux for VME and PC platform for DAS on Tore Supra. In 2011, the Tore Supra DAS dedicated software was ported on a Linux diskless PCI platform. The new generation was successfully tested during real plasma experiments on one diagnostic, called DCEDRE.

INTRODUCTION

The Tore Supra data acquisition system (DAS) (Fig. 1) was designed in the early 1980s [1] and has considerably evolved since then. Three generations of data acquisition units still coexist. Totalizing now 50 diagnostics and subsystems, the data acquisition units are operated simultaneously. The first generation was based on Multibus systems with an Intel processor and RMX™ operating system. Since 1995, most of these systems have been replaced by VME bus systems equipped with Motorola PowerPC processor boards running a Lynx OS™ real-time OS to obtain a more powerful architecture. The third generation was developed to perform extensive data acquisition for infrared and visible video cameras that produce large amounts of data to handle. It was supported by industrial PC over a Microsoft Windows™ OS.

The first two generations, Multibus and VME bus systems, are diskless with real time OS (RTOS). Diskless systems are desirable for reliability and maintainability as they share common resources like kernel and file system. The advantages of these architectures are:

- a common kernel
- a common file system
- easy updating and backup
- reliability (no hard disk)
- can be installed closest to the detector
- insensitive to the magnetic field
- easy to deploy

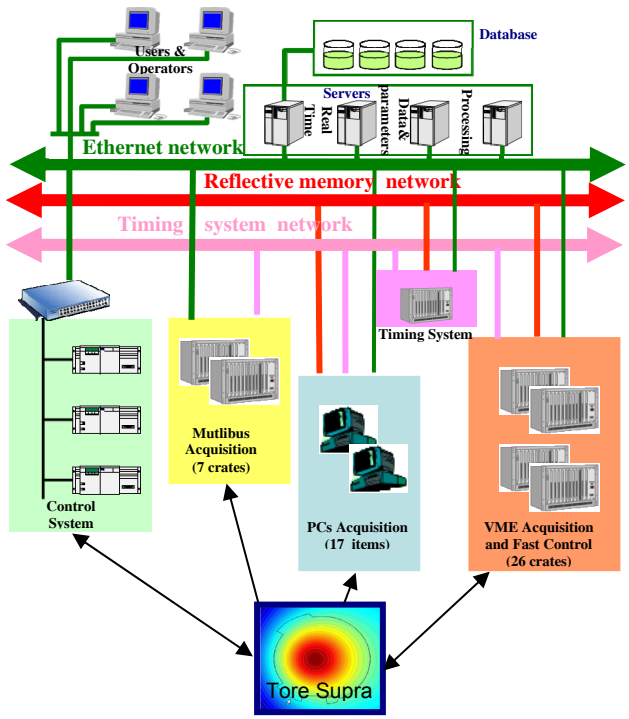


Figure 1: The Tore Supra data acquisition system.

These facilities considerably reduce the maintenance cost. Moreover, open source real-time operating system are now available that may provide free and convenient solutions for DAS. As a result, it was decided to explore an alternative solution based on an open source OS with a diskless system for the fourth generation. Currently, Linux OS is fairly mature to be used on DAS with pre-emptive and real time features on x86 architecture. In 2010, Linux distributions for PCI bus systems have been evaluated and compared to LynxOS real time OS. The latency and time response to any hardware interrupt has been achieved with x86 and x86 multi-core architecture target processors. The results obtained allowed to choose a version of Linux PCI platform for DAS on Tore Supra. These aspects and the new diagnostic DCEDRE are detailed in the following of the paper.

DAS NEEDS ON TORE SUPRA

Two kinds of DAS with different level of determinism are used on Tore Supra, hard real time systems and soft real time systems. The real time plasma control system

(hard real time system) on Tore Supra has a constraint of 2 ms cycle time. For example, one of Tore Supra safety components is provided by a spectrometer measuring the level of copper and iron [2]. The acquisition unit allows heating systems to adapt in real time the injected power into Tore Supra through a feedback on these levels of impurity. The acquisition unit must provide continuous values every 4ms. The data acquisition and computed results must be completed within this interval.

Soft real time systems must meet deadlines with a degree of flexibility. Data acquisitions that do not realize real time control accept missed dead lines like a read on data acquisition board. To solve this problem data are stored in acquisition card memory or directly into the computer memory by a DMA transfer between acquisition card and computer. The time constraint is 4 millisecond cycle time.

Data acquisition system must support Tore Supra standards on both hardware and software, especially the real time plasma control which is based on a reflective memory network with ScramNet card from Curtiss Wright. On software point of view, the message oriented middleware (MOM) RTworks [1], that is the heart of Tore Supra DAS must be supported.

NEW DAS FOR TORE SUPRA

New Hardware

A PCI bus platform has been chosen with CompactPCI (CPCI) standard for the new architecture (Fig. 2).

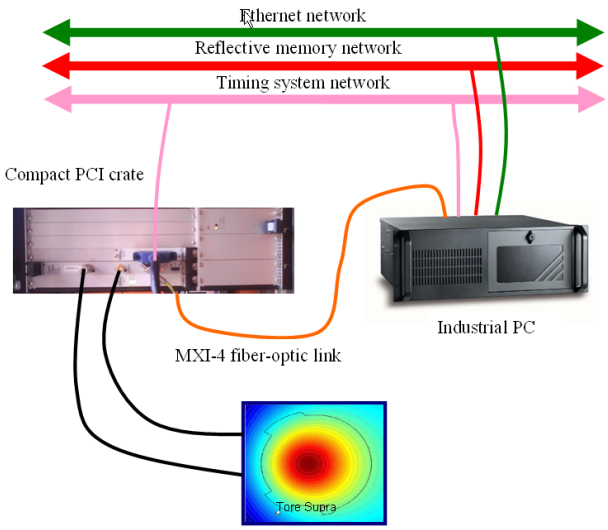


Figure 2: New DAS architecture.

CPCI bypasses the limit of five PCI slots in PC and offers a wide range of CPU and data acquisition boards. It is possible to use directly PXI cards, PCI and PMC (PCI mezzanine card) cards with adapter in a CPCI crate. The processor unit is not included in the CPCI crate. An industrial PC is connected to CPCI crate by a MXI-4 bridge from National Instruments. MXI-4 bridge provides a remote control of CPCI electronics cards from industrial PC through a fully transparent link connected with fiber-optic cabling. The separation of the I/O board from the industrial PC allows having a robust system like VME bus system and a lifespan of more than 10 years. This configuration has the advantage:

- easy upgrading of the calculation power without changing all the data acquisition
- a less noisy environment than for industrial PC for very fast data acquisition (2GHz).

For a reduced DAS development time, electronic cards providers are selected for hardware availability and drivers quality, so the development time is now down to half between design and delivery.

New OS, Linux

The first milestone was to find a Linux distribution. The target system for PCI bus system are x86 multi-core processor with industrial PC. X86 processor is widely used in Linux community which has led us not to use commercial distribution. A custom solution has been developed.

The main challenge will be to develop a diskless system. A diskless system on Tore Supra (named client) starts with Ethernet networks. Client has only memory and its file system is hosted on a server. The client-server architecture is showed on Figure 3. At startup, a client downloads the kernel on the server. The file system is common to all clients to facilitate the management. The user application development is realized on a compiler.

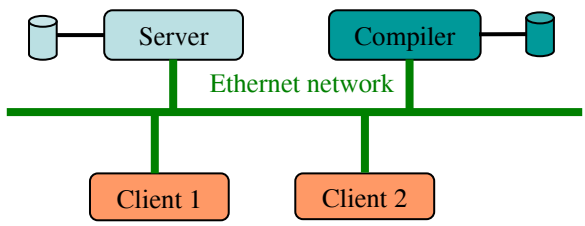


Figure 3: Client-server model on TS.

The target boot is based on PXE (Preboot eXecution Environment) boot. Some unnecessary services such as energy saving, virtual memory are disabled that slowdown the system. A hard RTOS solution will be evaluated in 2012 with Xenomai real time extension for Linux.

TESTS AND RESULTS

Tests were conducted to measure real time performance on few target systems and Linux version. Target systems tested were:

- X86 INTEL processor 2,8GHz.
- X86 INTEL dual core processor 3GHz on an industrial PC.

Different Linux version 2.6 with different preemption option were chosen to be evaluated:

- 2.6.28.10 preemptible kernel (low latency desktop)
- 2.6.26.8-rt16 Complete Preemption (Real-Time)

As referred in DAS needs on Tore Supra section a RTOS must meet deadline. Interrupt response time measurement and polling device can evaluate RTOS performance for a given target. Performance tests were also done with Lynx OS on a VME target system to compare Linux to a commercial RTOS.

The interrupt response time (Fig. 4) is the interval between the time when a hardware interrupt occur and the time when the user task wake up to run. The interrupt response time is the sum of interrupt latency, interrupt processing, context switching and task wake up.

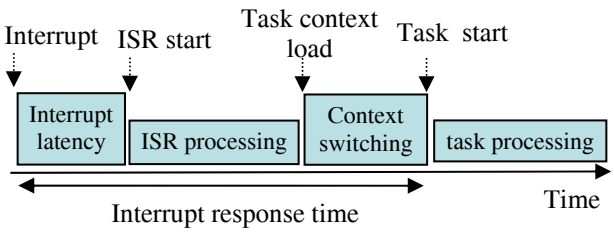


Figure 4: Interrupt response time definition.

For response time tests, user task simulates a processing consuming 100% of processing power to measure the impact on Linux to evaluate RTOS closest to reality. A 500µs interrupt frequency and 400µs processing simulation were used for this test during 1 hour.

Linux 2.6.28.10 on x86 dual core target is not far of hard RTOS Lynx OS (Fig. 5). Linux 2.6.28.10 is better in interrupt management than Linux 2.6.26.8 RT. Linux 2.6.28.1 interrupt time response is 38µs against 50µs for Linux 2.6.26.8 RT.

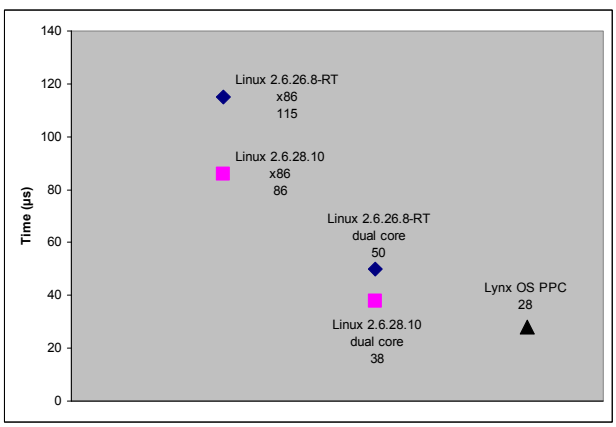


Figure 5: Interrupt time response result.

Some VME I/O (input/output) devices used on Lynx OS target do not generate interrupt to indicate that data are available. I/O devices require the application to generate the necessary request in order to interact with them. The application runs in a periodic loop and makes a request every time through the loop, called polling device. Polling is also a way to consider more quickly an event than interrupt. A test was done to find the polling frequency limit on each target system and Linux version (Fig. 6).

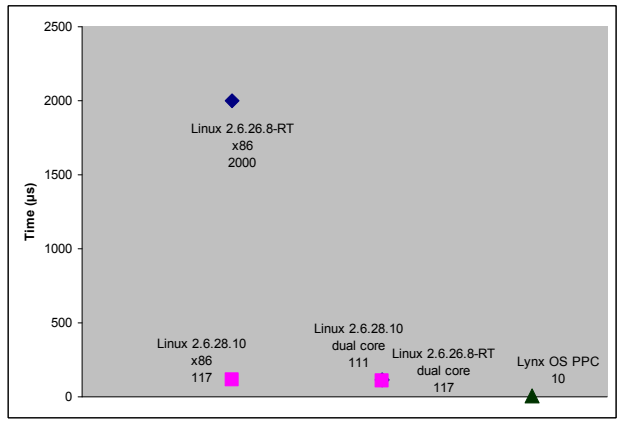


Figure 6: Polling frequency limit.

Lynx OS takes advantage on Linux. Moreover Lynx OS is completely freeze and do not reply to any request unlike to Linux. Linux 2.6.28.10 have a large advantage on x86 while both versions of Linux are equal on dual core. With upper CPU frequency than PPC, x86 dual core is close to hard RTOS performance.

NEW DIAGNOSTICS WITH LINUX

Linux with a system disk has been used successfully on a new diagnostic for the dust detection in 2010. It can count and quantify the dust sucked into a vacuum duct continuously 24/7. In 2011, the Tore Supra DAS custom software was ported on a Linux diskless PCI platform. The new generation was validated on plasma experiment for the so called DCEDRE diagnostic (Fig. 7).

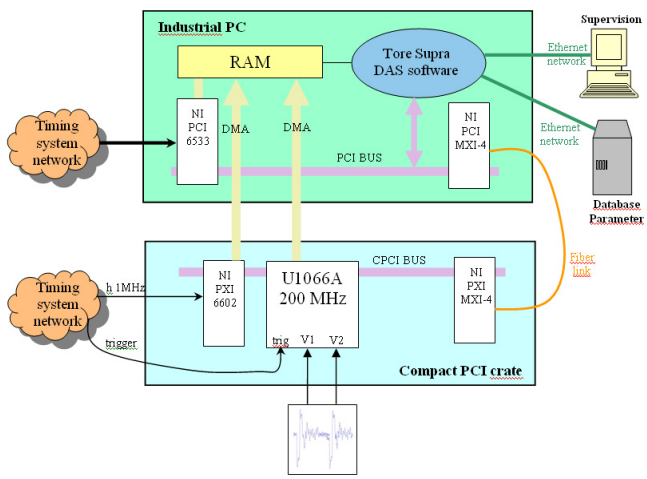


Figure 7: DCEDRE architecture.

This new diagnostic aiming at measuring the full frequency spectrum (0-200MHz) of electrostatic probes signal during high frequency power injection into the plasma gave preliminary results during Ion Cyclotron Range of Frequency (f=57MHz) heating.

Two signals are acquired on an Agilent U1066A, 200 MHz digitizer, 12 bit resolution with a wide bandwidth of 100MHz. An external trigger for timing system network starts data acquisition for 100000 samples at 200MHz samples rate. Data are time stamped with NI PXI 6602 card clocked by 1MHz timing system network.

Spectrum is obtained by Fourier transform of a set of 10^5 measurements (Δt=5ns). When the antenna is powered (P=1MW), peaks are obtained at f but also in some cases at f/2 (Fig. 8), giving some evidence of the complex interaction of the plasma edge, where the probe is located, with the wave.

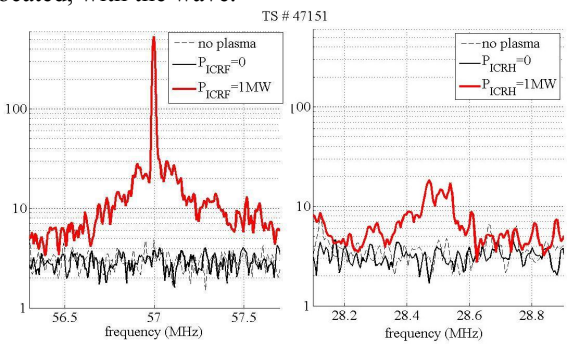


Figure 8: Blow-up of the power spectrum during ICRF heating experiment (acquisition rate: 200MHz).

CONCLUSION

This study showed that real-time Linux solutions can be used for the data acquisition systems of Tore Supra. Linux on PCI bus systems fulfils Tore Supra needs for soft real time application. Linux 2.6.28.10 preemptible kernel (low latency desktop) on x86 dual core is the best solution and obtain equivalent performance to Lynx OS on interrupt time response. PCI platforms will reduce the cost of hardware, development time and operating costs.

New Tore Supra DAS architecture is like ITER recommendation for ITER CODAC control system [3]. New DAS will be used easily for testing plant system controllers for ITER CODAC on Tore Supra experiments.

In 2011, the new diagnostic DCEDRE has validated the new diskless PCI architecture with Linux OS on experiments.

The next steps are:

- Performance tests with Xenomai in 2012.
- New diagnostics for plasma control with Linux 2.6 diskless system in 2012.
- Upgrade data acquisition unit of hybrid heating system on Linux diskless system with real time arc detection algorithm 100µs cycle in 2012.
- For x86 dual core system, to dedicate a core to Linux and the other to Xenomai.

REFERENCES

[1] B. Guillerminet et al., Tore supra continuous acquisition system. Fusion Eng. Design, 60, 427-434 (2002).

[2] F. Leroux et al., Real time data acquisition system for control and continuous acquisition in Tore Supra, Proceed. 13th IEEE-NPSS Real Time Conf., Montreal (2003).

[3] ITER CODAC Guideline for Fast Controlers, I/O Bus Systems and Communication Methods between Interconnected Systems January 2010.

LIA-2 POWER SUPPLY CONTROL SYSTEM

A. Panov, P. Bak, D. Bolkhovityanov
The Budker Institute of Nuclear Physics, Novosibirsk, Russia

Abstract

LIA-2 is an electron Linear Induction Accelerator designed in Budker INP as an injector for full scale 20MeV Linear Induction Acceleratorfor X-ray flash radiography with high space resolution. Inductors get power from 48 modulators, grouped by 6 in 8 racks. Each modulator includes three control sub-devices connected via internal CAN-bus to an embedded modulator controller, which operates under Keil RTX real-time OS control. Each rack includes a cPCI crate equipped with x86-compatible processor board running Linux with CX-server. Modulator controllers are connected to cPCI crate via external CAN-bus and interact with CX-server via extended KOZAK standard. In modulator controller operation many software mechanisms are used e.g. different modes, detection of an internal sub-device offline state, request repost, addressing, system error and other.

LIA-2

LIA-2 is originally an electron linear induction injector for 20MeV Linear Induction Accelerator designed in Budker INP for X-ray flash radiographywith high space resolution. It is a facility producing pulsed electron beam with energy 2MeV, current 1 kA and spot size less than 2mm. Beam quality and reliability of facility are required for radiography experiments. Accelerating section consist of 96 inductors based on the amorphous ferromagnetic laminated cores. Each modulator feeds two inductors, total 48. All modulators grouped by 6 in 8 racks.

Basic LIA-2 parameters are presented in Table 1. The accelerator includes electron beam forming, accelerating and focusing system, high voltage pulsed power system, beam diagnostic system, high vacuum system and control system.

Table 1: LIA-2 Basic Parameters

Parameter (Units)	Value
Maximum electron beam energy (MeV)	2.0
Maximum electron beam current (kA)	2.0
Number of pulses in the burst	2
Cathode heater DC power (kW)	2.5
Time interval between pulses in the burst (μs)	2 - 10
Pulse duration, flat top ±4% (ns)	200
Maximum repetition rate (Hz)	0.1
Min. beam spot size FWHM on the target (mm)	1.5

MODULATOR

Pulsed HV power supply system consists of 48 identical double pulse modulators, two charging units (one for each pulse) and coaxial feeding lines (20 cables for one modulator). Each modulator feeds two inductors in parallel and includes two Pulse Forming Networks (PFN) (one for each pulse), two cold cathode thyratrons TPI 1-10K/50 with a switched current 10 kA and working voltage 50 kV[1], two thyratron starters and a degausser (see Fig.1). Two starters (STR1/2) and degausser (DEG) are driven by Local Control Boards (LCB). In addition, each modulator is equipped by the controller (MC). 48 modulator controller, 144 LCB, 8 cPCI crates with CX-server and main control panel form HV power supply control system.

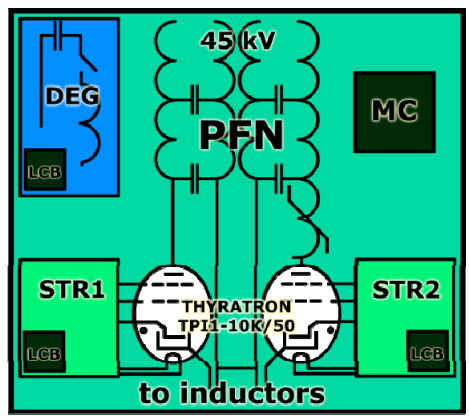


Figure 1: Modulator structure.

Hardware

Modulator controller is based on LPC2119 microcontroller by NXP. This microcontroller includes a 32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, together with 128 kB of embedded high-speed flash memory. The LPC2119 contains various 32-bit timers, 4-channel 10-bit ADC, two advanced CAN channels, PWM channels and 46 fast GPIO lines with up to nine external interrupt pins. The microcontroller performs following tasks: communication with external server and internal sub-devices on CAN-bus (125kb/s), slow interlocks gathering (response time less than 60 ms) and processing, parameters calibrations, indicator control (on Serial Peripheral Interface (SPI)), internal sub-devices supply control and other. Modulator controller also includes Programmable Logic Device (PLD) EPM3128 by Altera. This PLD is based on MAX3000A architecture and contains 128 macrocells. The EPM3128 performs fast interlocks gathering and processing with response time less than 1 us. Modulator

controller contains five inputs for fast internal interlocks with optic isolators, three inputs for synchronization pulses and output for final interlock with optic isolator. Inner structure of modulator controller with local control boards are presented on Fig.2.

Local Control Board is based on T89C51CC01 microcontroller by Atmel. This microcontroller implements 8-bit 80C51 CPU. The T89C51CC01 contains 32 KB flash memory and 2KB EEPROM, three 16-bit timers, 8 multiplexed inputs of 10-bit ADC, 5 outputs 8-bit PWM, one CAN channel and 34 digital I/O lines.

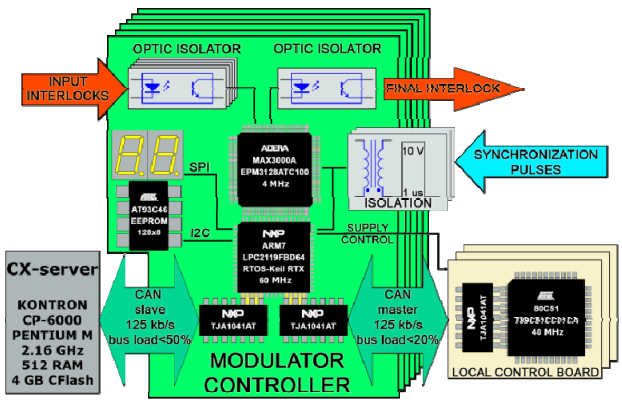


Figure 2: Modulator Controller and Local Control Boards.

Software

The modulator controller works under control of a Real-Time Operating System (RTOS) Keil RTX. The Keil RTX is a royalty-free, deterministic RTOS designed for ARM and Cortex-M devices. It allows creating programs that simultaneously perform multiple functions and helps create applications which are better structured and more easily maintained.

Features

- Royalty-free, deterministic RTOS with source code.
- Flexible Scheduling: round-robin, pre-emptive, and collaborative.
- High-Speed real-time operation with low interrupt latency.
- Small footprint for resource constrained systems.
- Unlimited number of tasks each with 254 priority levels.
- Unlimited number of mailboxes, semaphores, mutexes, and timers.
- Support for multithreading and thread-safe operation.

In the modulator controller Keil RTX is used with following features.

- Main clock 60 MHz
- System clock 1 kHz
- Round-Robin task switching
- Number of concurrent running tasks 20
- Task stack size 200 bytes

Main cycles

- Watchdog check 800 ms
- Interlocks check 50 ms
- Indicator refresh 300 ms
- Settings saving 1 m
- Internal management 45 ms

Modulator controllers are connected to cPCI crate via external CAN-bus and interact with CX-server [2] via extended KOZAK standard [3].

Addressing

Each modulator has unique functional address which sits in CAN connector. Based on this functional address modulator controller identify its network ID and physical address. Network ID is used in communication with CX-server and physical address is continually displayed on the front indicator. Physical address consists of a figure and a character. The figure indicates modulator rack serial number and the character indicates modulator serial number in this rack. All modulators with even character have even network ID. If one wishes to replace a modulator, one just removes old modulator and inserts a new one, than modulator identifies all addresses by itself.

Offline state detection and request age

One of the most important mechanisms is detection of an internal sub-device offline state. There are four counters (three individual and one shared) that define current state of internal sub-devices. Sub-device sits in offline state if counter value is less than minimum value and it sits in online state if counter value is greater than maximum value. Values of these counters are changed when internal communication task running. Counter value is decreased when the sender in a request processing task is recognized. Value of the shared counter will increase when there will be no more messages in a specified time domain (timeout for internal receive). Values of individual counters will increase when individual message will be sent for suitable sub-device. Also multi-request about current sub-devices states is sent if shared counter value is less than maximum level. Knowledge about current states of the internal sub-devices is especially important if sub-device sits in offline state and there are several messages for him. In this case all requests for this sub-device aren't sent but remembered.

All requests have an age counter. Value of the age counter is started to increase only when request is sent for suitable sub-device. Request gets old and then will be added in FIFO for sending if age counter value of this request is greater than maximum level. This mechanism is worked only when internal sub-device is online, but multi-request is sent until all sub-devices get offline.

Modes

Each modulator can be in one of six modes (*Sleep, Experiment, Action, Adjustment, Programming* and *Emergency* - ***SEAAPE***). On the Fig. 3 possible transitions from one mode to another are presented. Essentially one mode differs from another by Mask of Admissible Commands (MAC) which is used at the request reception.

Sleep is an initial modulator mode. All internal sub-devices of the modulator are switched off. In this mode operator can write minor settings (e.g. heating time of a hydrogen thyratron, masks of the enabled internal interlocks etc.). Final interlock is active.

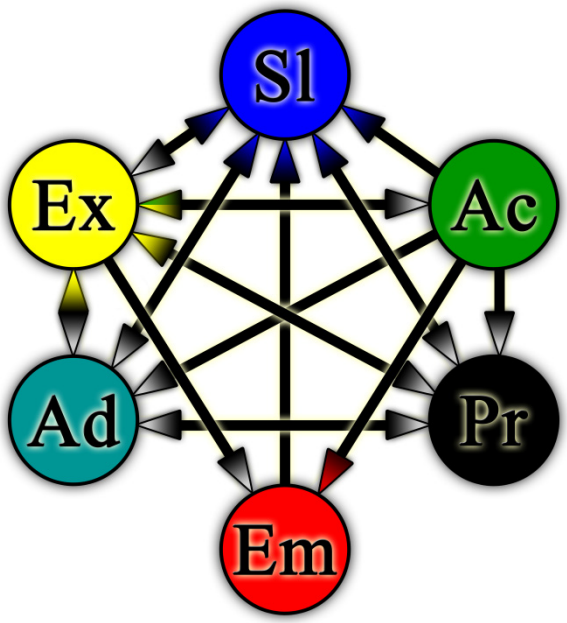


Figure 3: Possible transitions from one mode to another.

Button or power reset
Watchdog timer reset
Main shot
Enabled interlock
Command

Experiment is a basic modulator mode. All internal sub-devices of the modulator are switched on. In this mode operator can write major settings (e.g. heating current of a hydrogen thyratron, discharge arc current, degausser voltage etc.) and minor settings (like in *Sleep* mode). Final interlock is managed by current enabled internal interlocks. This mode is necessary for test LIA-2 shots. It is necessary to perform approximately one hundred shots in order for final main shot to be performed without any problems, because experiment with main shot is very expensive.

Action is a mode of main shot. In this mode operator cannot change any settings. If any enabled internal interlock appears then final interlock would active and modulator would change mode on *Emergency*. After main shot modulator change mode to *Experiment* by itself.

Adjustment is a mode for calibration of measured signals and other auxiliary procedures. Final interlock is active.

Programming is a mode for firmware download to the microcontroller. Final interlock is active.

Emergency is a mode when there is enabled internal interlock. In this mode all internal sub-devices are switched off, interlocks are hold and final interlock is active. Operator should change mode to *Sleep* for exit from this mode or reset microcontroller.

Modes approach makes all system more fail-safe and decrease human factor.

REFERENCES

[1] V. Bochkov Application of TPI-Thyratrons in a Double-Pulse Mode Power Modulator with Inductive-Resistive Load / V. Bochkov, D. Bochkov, A. Akimov, P. Logachev, V. Dyagilev, V. Ushich // IEEE Transactions on Dielectrics and Electrical Insulation, Vol. 17, Issue 3, pp. 718-722, June 2010.

[2] D. Bolkhovityanov "UI-oriented Approach for Building Modular Control Programs in VEPP-5 Control System", Proc. PCaPAC-2006, Newport News, VA USA, October 2006.

[3] http://www.inp.nsk.su/~kozak/designs/designs.htm

DETECTOR CONTROL SYSTEM OF THE ATLAS TILE CALORIMETER

G. Ribeiro, H. Santos, F. Vinagre, LIP, Portugal
S. Nemecek, Czech Republic Academy of Sciences, Institute of Physics, Czech Republic
G. Arabidze, Michigan State University, U.S.A
P. Lafarguette, Université Blaise Pascal, France

Abstract

This paper describes the development and implementation of the Detector Control System (DCS) of the Tile Calorimeter detector. The DCS must ensure coherent and safe operation of the Detector. It provides control and monitoring of all parameters of the system and gives to the user a comprehensive picture of the detector behaviour.

INTRODUCTION

The Tile Calorimeter [1] is one of the sub-detectors of the ATLAS experiment [2]. It is a sampling calorimeter made of steel plates (absorber) and scintillating tiles (active material). The design, general features and expected performance of the calorimeter are well described in the "ATLAS Tile Calorimeter Technical Design Report" [3]. The Tile Calorimeter consists of one barrel and two extended barrel parts. All the three sections have a cylindrical structure further sub-divided into 64 independent modules. The calorimeter cells are defined by grouping together sets of optical fibbers into bunches leading to photomultiplier tubes (PMTs). The front-end electronics and PMTs are located in the outer side of the Tile Calorimeter modules, in so-called electronics drawers.

The Detector Control System (DCS) is responsible for safe and coherent detector operation. All ATLAS sub-detectors have their own local DCS, which detailed architecture strongly depends on the structure of the general DCS system of the ATLAS experiment [4] and on electronics architecture and mechanical issues of the sub-detector itself. Each local DCS controls and monitors the operation of a sub-detector and related equipment.

Although each sub-detector is responsible for the implementation and for internal organization of the sub-systems, they must fully comply with the requirements defined by ATLAS central DCS [5]. The DCS of the sub-detectors must follow the general ATLAS DCS system architecture as much as possible unless there are special requirements where the sub-detectors need tailored solutions.

The DCS provides control and monitoring of the main systems of the Tile Calorimeter detector, which are the High Voltage distribution system and the Low Voltage Power Supply (LVPS) system. In addition, DCS is also responsible for interactions with detector calibration and data acquisition systems, and monitoring the detector infrastructure related systems: detector water-cooling and rack control.

TILE CALORIMETER DCS

The commercial Supervisory Control and Data Acquisition (SCADA) package PVSS II has been chosen by the Joint COntrols Project (JCOP) at CERN to implement Back End (BE) software for all LHC experiments [6]. The PVSS II is a commercial product, from Austrian company ETM. It is used to connect to hardware devices, acquire data from them, monitor their behaviour and to initialize, configure and operate them. PVSS II has a highly distributed and flexible architecture, and it allows connection of several autonomous systems through the network.

The BE system of the ATLAS experiment is organized hierarchically in three layers or levels as shown in Figure 1. This hierarchy allows the experiment to be divided into independent partitions, which have the ability to operate in standalone or integrated mode.

At the top layer, there are Global Control Stations (GCS), which are in charge of overall operation of the detector. They provide high level monitoring and control of all sub-detectors, while data processing and command execution are handled at the lower levels. The GCS is able to access all stations in the hierarchy.

The Sub-detector Control Station (SCS) represents the middle level of the hierarchy. The Tile Calorimeter, as a sub-detector of ATLAS, has its own SCS, which allows the complete operation of the sub-detector, by means of dedicated graphical interfaces. At this level of hierarchy, the connection with the TDAQ system, calibration systems and detector infrastructure takes place in order to ensure that detector operation and physics data taking are synchronized.

At the bottom level of the hierarchy are the Local Control Stations (LCS), which handles the low level monitoring and control of LV and HV systems of sub-detector. The LCS executes the commands received from the layers above.

In order to implement the BE system of the Tile Calorimeter DCS, five rack-mounted computers are used, located in USA15 racks. As it is shown in Figure 1, four of those are used as LCS stations (each for one Tile Calorimeter partition) and one as the SCS station. The operating system of those computers is Windows XP and they run PVSS II as a system service.

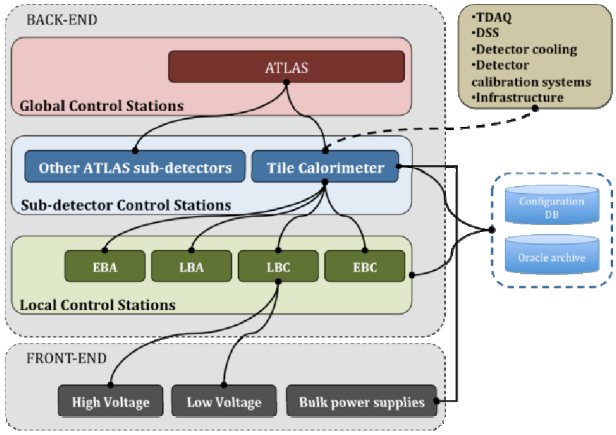


Figure 1: Hierarchy of the DCS of the Tile Calorimeter, as part of the ATLAS control system.

THE LOW VOLTAGE POWER SUPPLIES (LVPS) SYSTEM

The LVPS system is a two stage system: first stage converts 400V AC input into 200V DC output (100 m from the detector in USA15) and then second stage placed on the detector converts 200V DC into 8 independent levels of lower voltages in the range (-15V; +15V). These voltages are used to power the detector Front End (FE) electronics. The Tile Calorimeter FE electronics consists of digital and analog components of the readout system and High Voltage (HV) distributor system. In this section, the hardware entities of LVPS system and their interconnections are briefly described.

The LVPS system is composed by three devices: finger Low Voltage Power Supplies (fLVPS) located at the FE electronics of Tile Calorimeter, auxiliary boards (AUX boards) located in the racks of USA 15 and bulk power supplies providing 200V DC located also in the racks of USA 15.

The fLVPS and AUX board devices make use of Embedded Local Monitoring Board (ELMB) [7], as a general purpose I/O and processing unit for CAN communication [8]. The ELMB fully implements the industry standard of CANbus protocol and it provides minimal functionality of a slave node according to this protocol specifications.

The communication between the fLVPS and the Aux Board devices is made using the ELMB motherboard and the interface of the ELMB with the PVSS is done by an OPC server/client approach [9] where the client is provided by the PVSS manager and the OPC server by a software developed by the ATLAS DCS Central Team, CANopen OPC server. [10]

Figure 2 shows the layout of the ELMB based readout chain, for one Tile Calorimeter partition. In the LVPS system the maximum number of ELMB nodes per CAN branch is 16 and the number of CAN branches per partition is 5. Four of the CAN branches are used for communication with fLVPS devices and one for communication with AUX boards. The CAN Power Supply Unit (CAN PSU) is used to feed the CAN Transceiver part of the ELMB. The length of CAN branches, used to communicate with fLVPS devices is 120-150m and for AUX board devices it is ~10m.

The CAN communication speed for AUX board and fLVPS devices is set to 125KB/sec. Used CAN node addresses are from 1 to 16, as labelled on Figure 2. Branch #0 is used for the communication with AUX board devices and the others (branches #1 - #4) for the fLVPS devices. The numbering of Kvaser card [11] port matches to the branch numbering.

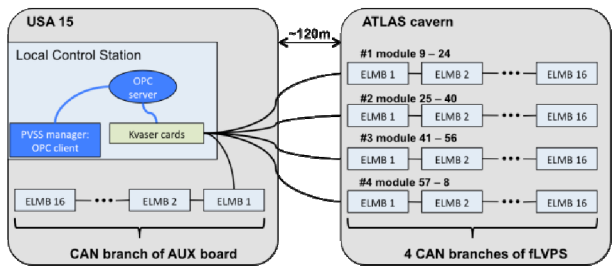


Figure 2: Communication schema with Aux Boards and fLVPS devices, for one Tile Calorimeter partition.

THE HIGH VOLTAGE DISTRIBUTOR SYSTEM

The High Voltage (HV) Distributor system is described in details [1], in the following we provide general information about the system power and communication lines at the ATLAS cavern.

The HV Distribution system consists of two devices: HV Bulk Power Supply (HV Bulk PS) and Super Drawer device. The HV Bulk PS devices are located in the ATLAS electronics cavern, called USA15 and the Super Drawer devices are located at the sub-detector area.

The HV Distributor system uses the HV Bulk PS device channels to provide input high voltage for each Super Drawer device. The HV Bulk PS provides 850 or 930 V, with the maximum DC current of 20mA. To use one input high voltage per Super Drawer, and to distribute and to regulate in-situ voltages of each individual PMT channel, with the precision better than 0.5V. The typical length of supply lines between HV Bulk PS and Super Drawer devices is about 120m.

The communication between the HV Bulk PS that uses ModBus/RTU protocol [12], and the DCS computer running PVSS which can only support ModBus/TCT, is achieved using a Port Server converter [13].

Inside the Super Drawer we have one HV_MICRO card, two HV_OPTO cards, HV internal and external buses, and the Flexible Bus (which links two HV Buses). The HV_MICRO controller card manages the voltages for individual PMTs through the HV_OPTO distributor cards and is used as the I/O and processing unit for CAN

communication providing functionality of the slave node according to the specifications of this protocol, as can be seen on Figure 3.

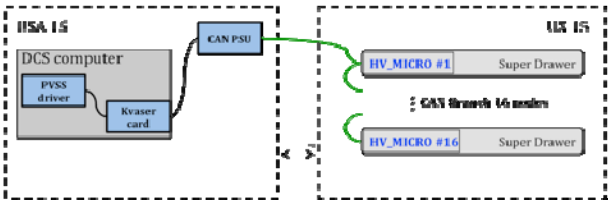


Figure 3: The HVMicro based readout chain.

CONCLUSIONS

The DCS implementation of Tile Calorimeter system follows requirements and suggestions from ATLAS DCS integration guidelines [5]. The DCS of the LVPS system represents comprehensive picture of the system and allows operator to have ability of full control over the system.

This paper provides detailed description of the hardware components of the LVPS system, emphasizes the critical parameters of the system and their monitoring thresholds for ALARM and WARNING. Implemented commands for individual device units of the LVPS system are given in details and time estimates for their execution are provided.

Usage of the Configuration and Conditions DB are also described in this paper. Analysis of the DCS data, from conditions DB showed that daily-recorded data size is in reasonable limit and allows understanding of the LVPS system behaviour. Implementation of the Configuration DB allows storing of the full information about the LVPS system calibration and nominal output voltages.

This paper presents a comprehensive picture of the Tile Calorimeter DCS system, as implemented following the requirements of the ATLAS DCS integration guidelines. The DCS of the LVPS system has been ready on time and proved to be both user-friendly and robust. It was successfully operated in a reliable manner for almost 2 years. A key element in this successful implementation was the correct selection of the building blocks since the beginning of the implementation.

We present the organization of the supervisory level for the LVPS system, information necessary for the operator to have the ability of full control over the system, and a detailed description of the hardware components of the LVPS system with emphasis on the critical parameters of the system and their monitoring thresholds for ALARM and WARNING. Details of the commands implemented for the individual device units of the LVPS system are given as well as time estimates for their execution.

Usage of the Configuration and Conditions DB are also described in this paper. Analysis of the DCS data from the Conditions DB showed that the daily recorded data is reasonable in size and allows understanding of the LVPS system behaviour. Implementation of the Configuration DB allows storing of the full information about the LVPS system calibration and nominal output voltages.

REFERENCES

[1] ATLAS Collaboration, Tile Calorimeter Technical Design Report, CERN/LHCC 96-42, CERN (1996)
[2]ATLAS Collaboration, The ATLAS experiment at the CERN Large Hadron Collider, JINST 3 S08003 (2008).
[3] ATLAS Collaboration, Atlas technical proposal, Technical Design Report, CERN (1994).
[4] Hierarchical Control of the ATLAS Experiment / Barriuso-Poy, Alex, CERN-THESIS-2007-037, Tarragona Univ., 2007.
[5] ATLAS DCS Integration Guidelines, ATLAS-DQ-ON-0013, EDMS Id: 685451, CERN (2007)
[6] A. Daneels and W. Salter. Selection and Evaluation of Commercial SCADA systems for the Controls of the CERN LHC Experiments. International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS (1999).
[7] H. Boterenbrood, B.I. Hallgren, The Development of Embedded Local Monitor Board (ELMB), ATL-DAQ-2003-053.
[8] CAN in Automation (CiA), www.can-cia.org.
[9] OPC, http://www.opcfoundation.org.
[10] CANopen Application Software for the ELMB128/H. Boterenbrood,http://www.nikhef.nl/pub/departments/ct/po/html/ELMB128/ELMB23.pdf, (2006).
[11] Kvaser, http://www.kvaser.com/.
[12] Modicom, Modbus Protocol, Reference Guide, PI-MBUS-300 Rev. J (1996).
[13] Port Server TS MEI is produced by Digi Internaltional (DigiBoard) company, www.digi.com.

NSLS-II BEAM POSITION MONITOR EMBEDDED PROCESSOR AND CONTROL SYSTEM*

Kiman Ha#, J. De Long, Young Hu, Yuke Tian, Joe Mead, Kurt Vetter and L. R Dalesio BNL
Photon Sciences, Upton, NY 11973, U.S.A.

Abstract

The NSLS-II is a 3 Gev 3rd generation light source that is currently under construction. A sub-micron Digital Beam Position Monitor (DBPM) system consisting of hardware electronics, an embedded software processor and EPICS IOC has been successfully developed and tested in the ALS storage ring and BNL Lab.

INTRODUCTION

The BPM is a key component for the NSLS-II machine, it was designed and development based on digital down conversion and FPGA digital signal processing technology. Performance testing has already shown 200 nm stability under controlled thermal conditions [1]. The Digital Front End (DFE) board utilizes a Virtex-6 FPGA from Xilinx, it provides a Gigabit Ethernet TEMAC core and a Micro-Blaze 32-bit RISC soft core processor for system on a chip embedded applications. Digital signal processing components are designed and implemented using Xilinx System Generator and the Matlab/Simulink tools. The embedded processor was designed using Xilinx EDK for kernel based multi threaded applications. The DBPM uses an EPICS IOC for control and monitoring. The DBPM communicates with the IOC using a simple, robust Ethernet protocol to directly accesses a 1 GB DDR-3 memory space. An Industrial PC running the Linux Operating System (Debian 6) supports waveform monitoring including; ADC raw data waveforms (117 MHz), Turn by Turn (TBT, 378 kHz) waveforms, Fast waveforms (FA, 10 kHz) and slow data (SA, 10 Hz). ADC raw, TBT and FA data is triggered on demand and supports a large 1 million sample history buffer (~ 32 Mbytes) transmitted to host computers (EPICS IOC, Matlab stand alone client computer). The EPICS IOC application program is based on the EPICS Asyn driver for custom TCP/IP protocol communications. It provides very stable communication with the Microblaze core. This paper will describe the NSLS-II new advanced DBPM functionality, embedded software and EPICS for control systems.

HARDWARE

Figure 1 shows the NSLS-II DBPM design for the Injection system. It contains separated DFE and AFE boards. The DFE board includes a Virtex-6 FPGA and Gigabit ethernet PHY chip and 128 Mbyte x 16-bit Flash memory. The AFE board has four individual RF receiver channels and high speed digitizer a PLL, attenuators and a pilot tone RF frequency synthesizer.

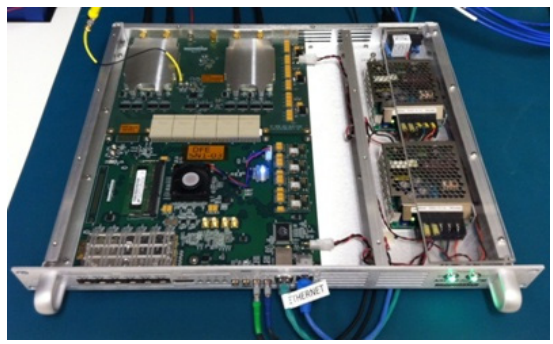


Figure 1: RF BPM for Injection System.

Embedded Event Receiver (EVR)

An embedded EVR with reduced functionality compared to the Micro Research Finland EVR board is implemented on the FPGA. The Embedded EVR synchronizes all of the BPM's data and generates synchronized clocks and trigger signals in addition to time stamps with 8 ns resolution.

Serial Data Interface (SDI)

A Fast Orbit Feedback System (FOFB) will be operational for the storage ring orbit stabilization. The NSLS-II requires a 10 kHz feedback rate. Vertical and horizontal Data is transferred to the cell controller every 100 uS using the Xilinx GTX transceivers at 5Gb/s. All data from all BPMs is collected at the cell controllers in approximately 20us.

RS232 Serial Interface

RS232 serial interface is used as a console port for the MicroBlaze processor. It is very helpful with system monitoring and diagnostics. The Microblaze initial boot information is displayed on the serial console. Boot manager, system reboot command and remote updating are operated using the serial console. This serial port connects with a terminal server (Ethernet to Serial converter box) for remote operation.

Tri-Mode Ethernet Interface

Virtex-6 Microblaze based TCP/IP socket communication performance is 4 Mbytes/sec, the RAW API mode is 3 times faster and the test results show a maximum of 12 MByte/sec. But the RAW API functions

* Work supported by DOE contract No: DE-AC02-98CH10886
#*kha@bnl.gov

have some limitations in that buffer overflow occurs during the multiple frame transmits. The DFE board utilizes onboard Marvel Alaska PHY device (88E1111) for Ethernet communications it supports 10/100/1000 Mbps/sec. The DFE board supports only GMII interface from the FPGA to the PHY chipset. The PHY connection to the EPICS IOC is through an RJ-45 Ethernet connector with built-in magnetics. The FPGA clock generator provides a 25 MHz external clock to PHY clock input.

EMBEDDED SOFTWARE

Figure 2 shows the internal FPGA components of the embedded processor and HDL signal processor. Xilinx provides a flexible embedded software development environment. The DBPM target board embedded software uses Xilinx Kernel mode. The Microblaze has application threads for multiple socket ports communicating with client applications. Multi sockets provide optimized communication speeds for different data structures. For example large waveform array reading (~32 Mb) takes approximately 30 seconds, to allow higher priority data transfers the large waveform thread can be pre-empted. For the embedded firmware development Xilinx software tool versions: XPS/SDK 13.2, Microblaze 8.2a, XilKernel 5.0 and LWIP 3.0 a are used..

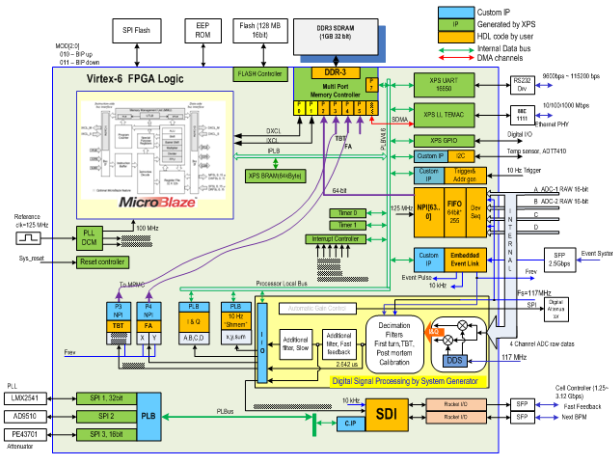


Figure 2: FPGA internal components for control and data processing.

MicroBlaze Soft Processor

The Microblaze core is a 32-bit RISC soft core processor using a Harvard architecture optimized for embedded applications. The Microblaze has a floating point unit, memory management unit, instruction and data caches and many other optimization functions. The EDK tool provides a simplified configuration wizard and it easily configures the BSP for FPGA embedded application. The Microblaze primary bus is the PLB, as master the MicroBlaze connects to slave PLB I/O peripherals. For the Virtex-6 DFE board the Microblaze and PLB are running at 100 MHz.

Lightweight IP (lwIP)

LwIP is an open source TCP/IP network stack for Microcontroller Ethernet applications. Xilinx EDK provides lwIP software customized to run on Xilinx Embedded systems. lwIP provides a TCP/IP socket layer API and raw API layer for applications. lwIP socket API is very similar to the Berkeley/BSD sockets therefore easily implements network programming . The lwIP v3.00.a version library is provided by Xilinx EDK 13.2.

MicroBlaze Processor Local Bus (PLB)

The Microblaze local bus interface is the 32-bit PLB and all I/O peripherals are PLB slaves. The Microblaze Data/Instruction cash bus interface to the multi-port memory controller which connects to the DDR-3 memory. Using external cash increases Microblaze and TCP/IP communication performance.

Multiport Memory Controller (MPMC)

Figure 3 and 4 show DDR write timing and the memory map. The MPMC is a key component for system performance tuning and real time operation. The MPMC physicaly interfaces to the DDR-3 memory and provides eight individual ports for the user bus interfaces. The DFE uses a SODIMM socket type 1GB DDR-3 memory and it is clocked at 400 MHz. It operates at double the rate of the clock and total the transfer rate is 32bits * 800 MHz. MPMC port 0 and port 1 is assigned for Microblaze Data/Instruction cash. Port 2 is a soft DMA port for the Gigabit Ethernet driver. Port 3 is connected to the ADC raw data path and operates at the ADC sampling frequency (117 MHz). Port 4 is assigned to the TBT data bus and Port 5 is assigned to 10 kHz FA data. One million points of ADC raw data and TBT and FA data can be copied to DDR-3 memory on demand or on a repetitive trigger.

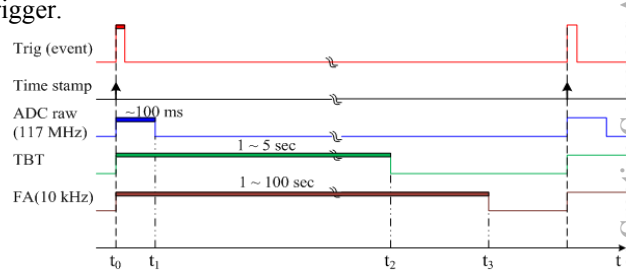


Figure 3: Timing of the waveform transmitter to DDR-3.

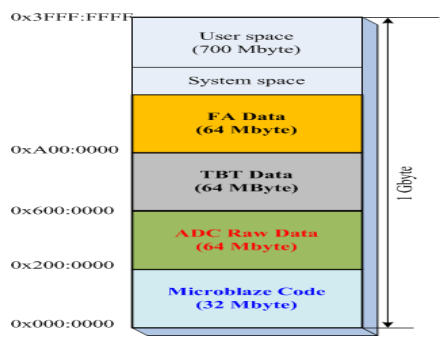


Figure 4: DDR-3 memory map (1 Gbyte).

BPM CONTROL AND IOC

Feedback control and event system links are separated from Ethernet because the global fast orbit feedback rate is 10 kHz and must provide global BPM data to the cell controllers deterministically. Similarly the event system runs at 125 MHz and requires determinism. Figure 5 shows a single cell BPM system configuration. The NSLS-II storage ring is divided into 30 such cells. Each cell has a VME EVR, a Linux IOC for communications and a cell controller for the fast orbit feedback system.

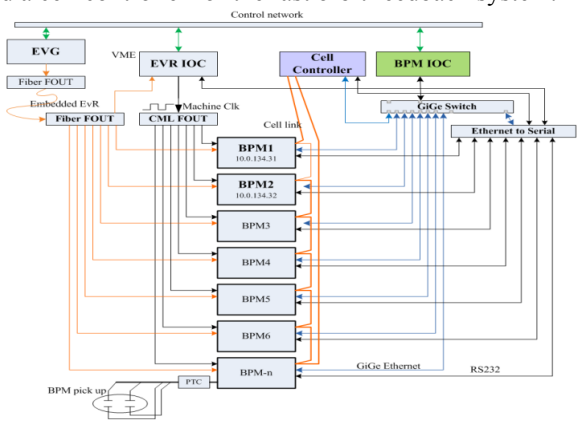


Figure 5: Layout of single cell.

Figure 6 shows embedded BPM software and EPICS IOC connect configurations. The Control System uses the Linux IOC for DBPM signal control and monitoring. IOC and BPM communication ports support Gb Ethernet. This communication uses a simple protocol based on register memory mapping. The DFE main device which is a Virtex-6 provides a hardware TEMAC Ethernet core and Microblaze soft processor that handle TCP/IP communication protocols. The EPICS IOC has three threads running for communication with the DBPM. The SA thread is designed for reading SA data (1024 bytes) and control register reads at the 10 Hz rate. The Microblaze SA server uses the RAW TCP/IP API functions for improved performance. The DDR waveform thread uses the socket TCP/IP API functions for stable communication because waveform data sets can be as long as 32 Mbytes. The DBPM EPICS IOC is an Industrial PC running the Linux Operating System, most communication is based on Ethernet ports.

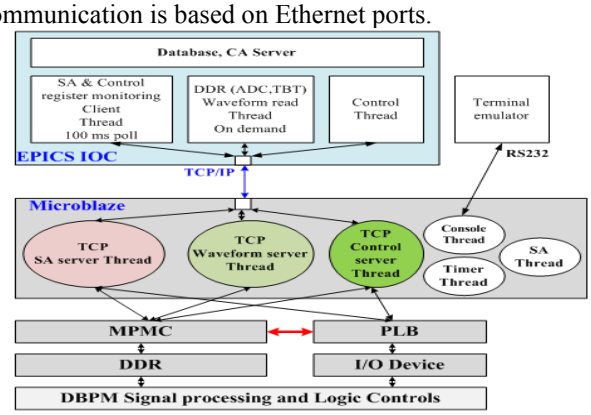


Figure 6: Control systems for BPM.

EPICS EXTENSIONS

Figure 7 shows EDM client examples for BPM controls and data measurements. ADC waveforms include 4 button signals: A, B, C, D and SUM data, the displayed waveform length is 4 k points and a slider PV can control the offset address within the 1Mpoint waveform. TBT waveforms include A, B, C, D, X and Y position, Sum and Q data. The TBT waveform length is 8000 points and a slider PV can control the turn offset within the 1 Million turn waveform. 10 kHz waveforms are the same as TBT waveforms. 10 Hz data is the 10 kHz data averaged over 1000 samples.

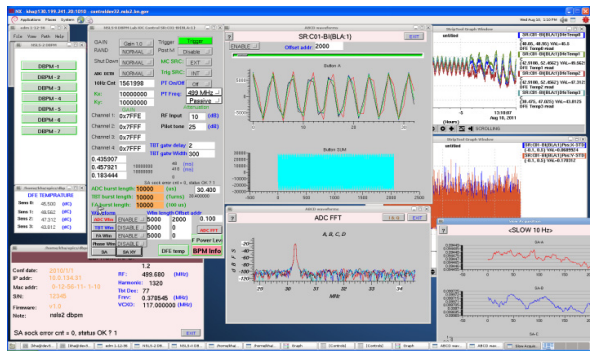


Figure 7: EPICS EDM screen for control and waveform monitoring.

SUMMARY

The development of sub-micron BPMs and their control system for the NSLS-II has been completed and successfully tested in the EPICS environment. FPGA based embedded software provides very reliable and stable operation. This year we will produce BPMs for the Linac and Injection systems, continue developing embedded firmware and EPICS software for the Booster and Storage ring BPMs.

ACKNOWEDGMENT

The authors are grateful and thank the NSLS-II beam diagnostics group. A special thanks to M. Maggipinto, A. Dellapenna Jr., and B. Bacha for their support. Also thanks to Michael Davidsaver and Jayesh Shah for VME event system support, Robert Petkus for BPM IOC and network integration.

REFERENCES

[1] K.Vetter et al., "NSLS-II RF Beam Position Monitor", BIW10, Santa Fe, NM, US.

A NEW FAST TRIGGERLESS ACQUISITION SYSTEM FOR LARGE DETECTOR ARRAYS

P. Mutti*, M. Jentschel, E. Ruiz-Martinez, J. Ratel, F. Rey, W. Urban,
Institut Laue-Langevin, Grenoble, France

Abstract

Presently a common characteristic trend in low and medium energy nuclear physics is to develop complex detector systems to form multi-detector arrays. The main objective of such an elaborated set-up is to obtain comprehensive information about all reactions. State-of-art γ-ray spectroscopy requires nowadays the use of large arrays of high purity Ge detectors (HPGe) often coupled with anti-Compton active shielding to reduce the ambient background. In view of this complexity, the front-end electronics must provide precise information about amplitude, time, detection position and possibly pulse shape. The large multiplicity of the detection system requires the capability to process the multitude of signals from many detectors, fast processing and very high throughput of more than 10^6 data words/sec. The possibility of handling such a complex system using traditional analogue electronics has shown rapidly its limitation due, first of all, to the non negligible cost per channel and, moreover, to the signal degradation associated with complex analogue path.

Nowadays, digital pulse processing systems are available, with performances, in terms of timing and energy resolution, equal if not better than the corresponding analogue ones for a fraction of the cost per channel. The presented system uses a combination of a 15-bit 100 MS/s digitizer with a PowerPC-based VME single board computer. Real-time processing algorithms have been developed to handle total event rates of more than 1 MHz, providing on-line display for single and coincidence events.

INTRODUCTION

The principle functions of the front end electronics for nuclear physics applications are to acquire the electrical charge pulses generated by a radiation detector, to extract the quantities of interest and to convert them into a digital format. Usually these quantities of interest are the particle energy, the time of arrival and the shape information [1].

In an ideal acquisition system, the analog signal has to be converted into a digital data stream as earliest as possible and then transferred to preserve the information. Today with the modern digitizers the restrictions of the A/D conversion have been reduced because of the increment of the sampling frequency and the resolution in terms of the number of bits. Besides the loss of information due to the A/D conversion error, the major problem of the fully digital approach is the huge amount of data to handle. It is almost impossible to have a system in which the data throughput rate allows the acquisition of all the channels and makes the on-line analysis at the same time, the loss of data it will be dramatic.

In order to reduce the amount of data to be transferred, a Field Programmable Gate Array (FPGA) can be used to perform on-line digital pulse processing and consequently extract and save the relevant quantities [2].

The present paper will start with the description of the chosen digital approach solution to maximize the throughput of the acquisition system. The digital pulse processing algorithm section summarizes the applied techniques to reduce the amount of data and extract the quantities of interest. It will be followed by a description of the on-line acquisition engine implemented at the Institut Laue-Langevin to cover the experimental needs. The achieved performances in a series of real experiments will be discussed followed by the most important conclusions gathered in the development of the project.

DIGITAL APPROACH

In an ideal acquisition system, the analog signal is converted into a digital data stream as earliest as possible and then transferred, without any loss of information, to a computer that can make the analysis and extract the quantities of interest off-line. Thanks to a great variety of hardware available, the end user is able to best match the requirements of the specific experiment concerning the critical phase of the A/D conversion. Applications implementing very fast response detectors or requiring precise time determination will most profit from a reduced number of bits (typically 8 or 10 bit) and a high sample frequency (from 500 MS/s up to 5 GS/s). The opposite is true when high-energy resolution is required but the detector signals are relatively slow. In such a case 14/15 bit digitizers are better suited at a price of a lower sampling (typically 100 MS/s). Besides the loss of information due to the A/D conversion error, the major problem of the fully digital approach is the huge amount of data to readout. It is necessary to restrict the acquisition to some selected parts of the signals.

Like in a common oscilloscope, this happens firstly by means of the trigger defining a certain time window in which the signal has to be recorded (acquisition window). The difference between a common oscilloscope and a digitizer is that the latter has several memory buffers for the triggers and can save subsequent acquisition windows without any dead time between them. Therefore, the acquisition can take place without the loss of any event, no matter what is the frequency and the distribution of them, at least until the readout rate allows the empting of the memory buffers. For the present application we have chosen a CAEN digitizer (mod. V1724 – 8 channels 15

*mutti@ill.eu

bit 100 MS/s). Each channel of the digitizer is equipped with a dedicated FPGA that continuously writes the digital samples coming from the ADC into a circular memory buffer. When it receives the trigger, it keeps saving the post trigger samples, then freezes that buffer (which is made available for the readout) and continues to save the samples into a new buffer. The record length (number of samples in the acquisition window) and the size of the post trigger are both programmable.

The use of the trigger is the first step towards the reduction of the overall data throughput rate. However, in most cases, this is not sufficient because the number of channels, the event rate (i.e. pulses per second) and the event size (record length for each pulse) are such that the total throughput rate exceeds the capability of any type of readout bus. It is hence necessary to apply some data reduction technique.

The simplest approach is to discard the channels or some parts of the records that do not contain any significant information (zero suppression). The ideal would be to apply on-line digital algorithms able to extract from the signal only the quantities of interest (i.e. the time of arrival or the energy of the particle) and save and transfer to the computer only those numbers. Sometimes, the best compromise is to have an intermediate solution in which the digitizer saves part of the waveform together with the quantities calculated on-line.

DIGITAL PULSE PROCESSING ALGORITHM

The purpose of the Digital Pulse Processing (DPP) is to perform on-line signal processing able to transform the row sequence of samples into a compressed data packet that preserves the relevant information. For this purpose, a specific algorithm implemented in each channel's FPGA allows to extract the energy and/or the arrival time of the detected pulses. The pulse conversion starts as soon as a trigger is generated. In our case, this is via a voltage step overtaking a programmable digital threshold rather then comparing the absolute voltage level, as in the case of analog systems.

For an accurate time determination, the traditional approach would make use of a Constant Fraction Discriminator (CFD), where the zero crossing of the sum of the input and the delayed, attenuated and inverted input, delivers the wanted information independently of the amplitude of the pulse. In the digital CFD the FPGA calculate the second derivative of the input signal (the direct output of the detector preamplifier). The input signal can be represented as the sum of two exponential decays being the fast component determined by the time constant of the detector and the slow one by the RC of the preamplifier. A simple calculation demonstrates that the zero crossing of such a second derivate depends only on the two time constants while it does not depend on the pulse amplitude. The zero crossing of the second derivate of the input signal delivers, therefore, an accurate time determination with an uncertainty equivalent to one unit of the sampling time.

A trapezoidal filter is applied to the input signal to derive its amplitude, proportional to energy. This filter can be shortly described as an algorithm able to transform the typical exponential decay signal generated by a charge sensitive preamplifier into a trapezoid [3]. The height difference between the signal baseline and the flat top of the trapezoid is proportional to the amplitude of the initial pulse.

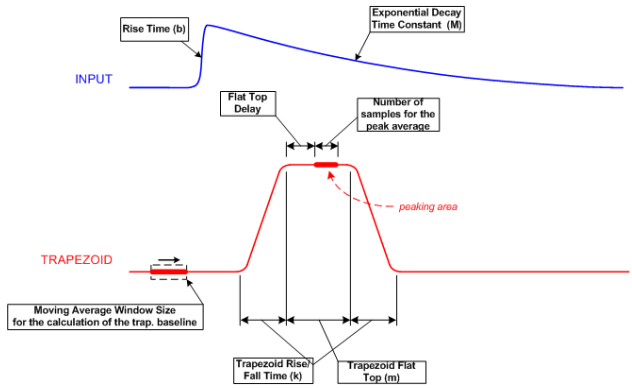


Figure 1: Trapezoidal filter with the relevant parameters.

Figure 1 describes all the relevant parameters necessary for a proper determination of the pulse height. This trapezoid plays more or less the same role of the shaping amplifier in a traditional analog acquisition system. We want to highlight the analogy between the two systems: both have a shaping time constant and must be calibrated for the pole-zero cancellation. For both, a long shaping time gives a better energy resolution but increases the probability of pile-up. Both are AC coupled with respect to the output of the preamplifier whose baseline is hence removed, but both have their own output DC offset, which constitutes another baseline for the peak detection. Setting the parameters of the trapezoidal filter is equivalent to change the shaping time in a classical spectroscopy amplifier.

ON-LINE ACQUISITION ENGINE

Our newly developed acquisition system consist of the following three main elements: a number of digitizers according to the required number of detectors, a processor card providing real-time algorithms and the instrument control software responsible for setting parameters, the real-time display and list-mode storage of data. Figure 2 represent schematically all the elements on the newly developed acquisition engine that will be discussed in further details in the following. The present system has been dimensioned according to the needs for the upcoming γ-spectroscopy campaign of measurements using the EXOGAM [4] high efficiency array of γ-ray detectors. Up to 10 segmented clover detectors will be used in the array. Segmentation is required to provide the optimum performance in terms of efficiency, energy resolution and minimization of multiple-hit events. These

Ge detectors will be surrounded by bismuth germanate BGO suppression shields and will also have passive heavy metal collimators for background reduction purposes.

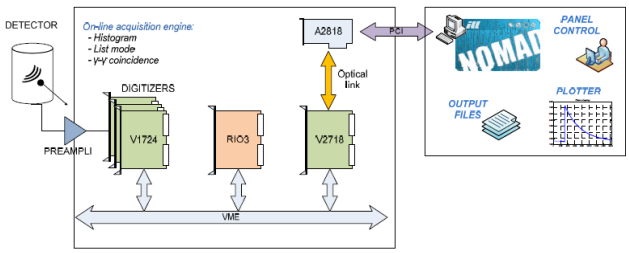


Figure 2: Schematic view of the acquisition engine.

Digitizers

The model used is the CAEN waveform digitizer board V1724. It is a 1-unit wide VME 6U module housing 8 channels 15 bit 100 MS/s flash ADC with threshold auto-trigger capabilities [5]. Trigger signals can be provided via the front panel input as well as via the VME bus, but they can also be generated internally, as soon as a programmable voltage threshold is reached. The individual auto-trigger of one channel can be propagated to the other channels and onto the front panel trigger output. The board features a front panel clock/reference I/O and a PLL circuit for clock synthesis from internal/external references. In this way, several boards can be phase synchronized to an external clock source or to a V1724 clock master board.

The digitizers consist of a common motherboard housing specific mezzanine cards for each channel. The motherboard houses the clock logic management and the readout control FPGA that provides the trigger management and acts as a bridge between the local bus and the different interfaces (VME 64X, Optical link, PCI express, USB 2.0). Mezzanine cards house the input front-ends and the channel FPGA that accounts for the analog-to-digital conversion, the digital pulse processing algorithm and the memory buffers. The readout control FPGA accesses the mezzanine cards through the local bus.

Processor Card

The real-time acquisition engine is implemented in a Power PC based VME board computer equipped with 256 MB of memory. This board does not contain any operating system to optimize real-time performances and prevent any dead time. A specific set of routines has been developed to account for the acquisition and all the functionalities necessary to access the digitizers [6]. We have developed real-time processing algorithms capable to handle total event rates of more than 1 MHz, providing on-line display for singles and coincidence events. The purposes of the processor card are:

- to create the data structures containing the variables and the parameters for the acquisition
- to initialize and program the digitizers
- to read the data from the digitizer, using BLT in the VME bus
- to analyze the acquired data extracting the wanted information and store them in the internal memory
- to generate the histograms for each input and for the coincidences.
- to create the list mode events
- to store list-mode data to the external mass storage.

Instrument Control Software

The instrument control software is acting as interface between the acquisition hardware and the user. It offers a full graphical interface to access all parameters related to the configuration of the digitizers as well as of the acquisition card. It allows real time visualization of all relevant spectra including double and triple coincidences.

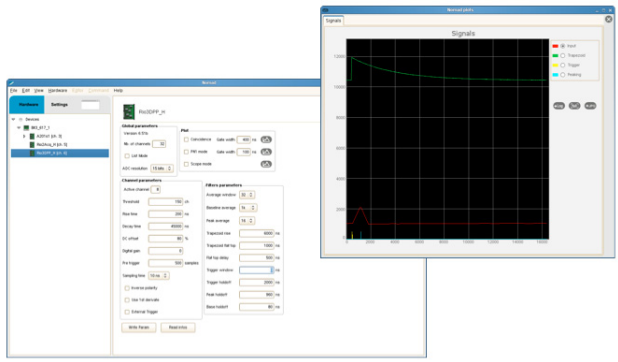


Figure 3: The digitizers setup screen and the oscilloscope live display.

Figure 3 depict the oscilloscope mode display used to tune the sampling parameters for each channel of the digitizer cards. This mode offers the unique advantage to visualize instantaneously the effects of any parameter modification with respect to all signals involved in the acquisition.

RESULTS

The new acquisition engine has been implemented with two different detector setups. The first test has been performed using an array of eight HPGe detectors for γ-spectroscopy studies from (n_{th}, γ) reaction. The goal of this experiment was, amongst others, to verify the beam quality, with respect to background and collimation, as well as the overall performance of the new acquisition in view of the upcoming EXOGAM campaign of measurements.

The detector setup of the second test was more complex, implementing a combination of HPGe and BGO detectors. The scientific goal of the experiment was to measure the e^+e^- pair-production cross-section close to the 1022 keV threshold [7]. We had the possibility to verify the accurate time synchronization amongst several digitizer cards as well as the capability of the system to handle detectors with completely different time response. Figure 4 shows the energy spectrum measured by the HPGe detector from ^{152}Eu radioactive source. We could

obtain an energy resolution of 2.45(5) keV at the 1408 keV γ-line. In the zoom area the doublet at 1085 and 1089 keV is clearly resolved.

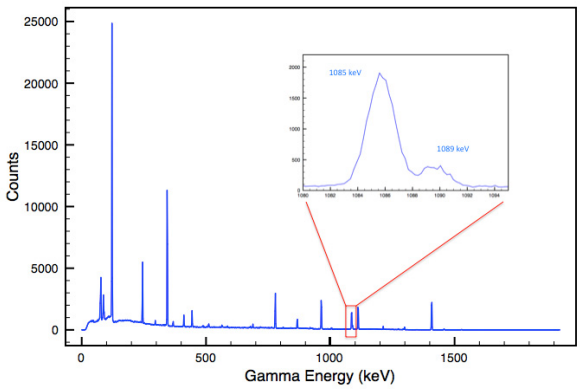


Figure 4: HPGe energy spectrum from the ^{152}Eu radioactive source.

From the analysis of the recorded list-mode data acquired using a ^{22}Na radioactive source in pair spectrometer mode, we could derive the timing properties of the new acquisition. The ^{22}Na source was selected because of the close vicinity of its 1274 keV γ-line with the energy range interesting for the experiment. Figure 5 reports the spectrum of the coincidence between BGOs and Ge detectors. The time difference between γ-detection in any of the BGO detectors and the corresponding detection in the Ge detector is plotted versus time. A single, very clean peak is visible at about 200 ns after the γ-detection in the BGOs.

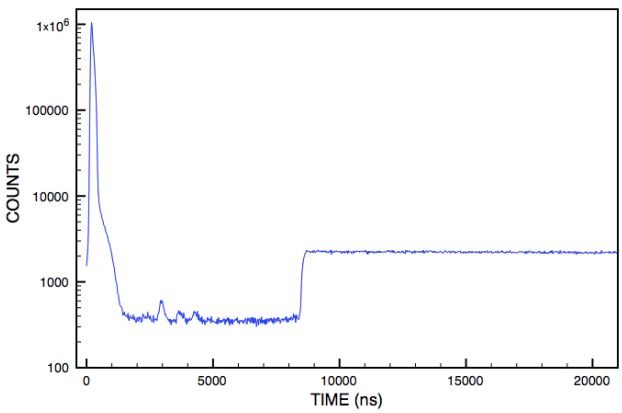


Figure 5: Time between BGO (start) and Ge (stop), measured for ^{22}Na source in pair spectrometer mode.

After about 8 μs the spectrum shows a rather low and extremely constant level of random coincidences. The absence of BGO-Ge coincidences below 8 μs could be explained by the conversion time of the system necessary to derive the energy and the time information. We do not have an evident explanation for the deformation of the main peak toward longer coincidence time. One possibility could be related to a slight misalignment in time on the signals from some of the BGOs.

CONCLUSIONS

The new acquisition system developed at the ILL implements a fully digital approach where the A/D conversion is performed directly on the preamplifier signal. With a reduced cost per channel, with respect to the analog approach, we are able to obtain information on energy, time and pulse shape. A direct comparison between the full digital system with an analog equivalent using the same detector did not show any significant loss in energy resolution.

All the relevant parameters are stored on EPROM for a maximum of flexibility and reproducibility. In the practical implementation using two different detector setups, the system has shown great modularity and the capability to be adapted to different types of signals from different detectors.

One major advantage is the possibility to acquire all signals independently from any trigger condition, reducing the chance of errors in the electronic setup and preserving a maximum of flexibility for the off-line analysis. Despite the large amount of data collected, the system never showed any significant dead time and our laboratory tests showed the possibility to achieve count rates higher that 1 MHz.

The combination of the new hardware and our control software offers the possibility to access a full set of graphical tools for configuration purposes as well as for on-line display of singles and coincidence spectra.

REFERENCES

[1] William R. Leo, “Techniques for Nuclear and Particle Physics Experiments A How-to Approach”, ISBN 3540173862, (1992).
[2] Carlo Tintori, “Digital Pulse Processing in Nuclear Physics”, CAEN Internal Report, (2010).
[3] V.T. Jordanov and G.F. Knoll, NIM A, 345 (1994).
[4] F. Azaiez, Nucl. Phys. A, 654 (1999).
[5] CAEN Technical Information Manual MOD. V1724 8 CHANNEL 14 BIT 100 MS/S DIGITIZER, (2010); http://www.caen.it.
[6] CES RIO3 8064 PowerPC-Based VMEx-LI Processor Board User Manual DOC 8064/UM Version 3.4; http://www.ces.ch.
[7] M. Jentschel, W. Urban, P. Mutti, P. Courtois, G.S. Simson and R. Frahm, Phys. Rev. C, in print (2011).

EVOLUTION OF THE CERN POWER CONVERTER FUNCTION GENERATOR/CONTROLLER FOR OPERATION IN FAST CYCLING ACCELERATORS

D. Calcoen, Q. King, P.F. Semanaz, CERN, Geneva, Switzerland

Abstract

Power converters in the LHC are controlled by the second generation of an embedded computer known as a Function Generator/Controller (FGC2). Following the success of this control system, new power converter installations at CERN will be based around an evolution of the design – a third generation called FGC3. The FGC3 will initially be used in the PS Booster and Linac4. This paper compares the hardware of the two generations of FGC and details the decisions made during the design of the FGC3.

FGC ORIGINS

LHC power converters are current sources composed of three elements:

- A *voltage source* consisting of the components needed to convert electrical energy from a given source into the desired voltage on a given load. This includes protections and analogue voltage regulation.
- Calibrated *current measurement* devices to acquire the measurement needed for the current regulation. In some cases these require very high levels of absolute accuracy, stability and very low noise.
- A *Function Generator/Controller (FGC)* which integrates the voltage source and current measurement system to become a current source by providing various services including function generation, current regulation and state control.

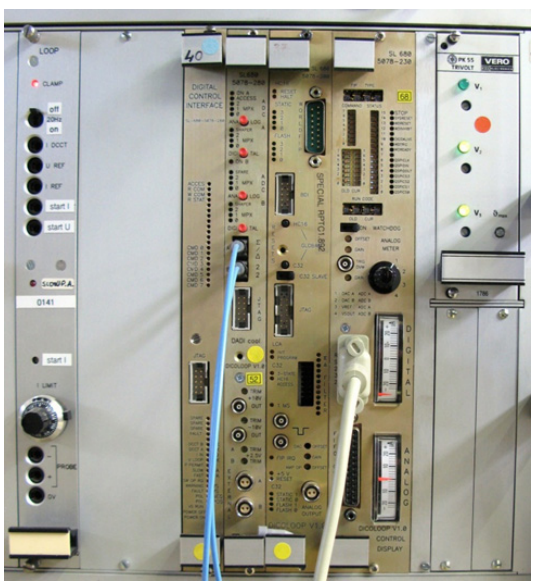


Figure 1: FGC1 developed from 1996-1999.

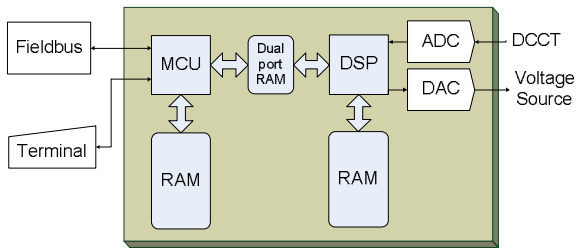


Figure 2: FGC core architecture.

The FGC project was launched in 1996 [1] with a specification that included:

- Resolution and tracking error down to 1 ppm on the current.
- Timing synchronisation to better than 1 ms.
- Radiation tolerance of up to about 2 gray per year with a life expectancy of 20 years.
- Use of WorldFIP as the synchronous fieldbus for remote operation and including support for real-time control at up to 50 Hz for accelerator-level regulation loops (i.e. orbit, tune and chromaticity).
- Mechanical and electronic design by CERN and manufacture of 2000 units by industry.

More specifically the FGC is responsible for:

- Generating the synchronised function of current versus time to be used as the reference, as required by operations at the system (accelerator) level.
- Measurement and regulation of the current.
- Control and monitoring of the power converter state.
- Integration with system level interlocks, in this case the LHC Powering Interlock Controller (PIC).
- Remote (WorldFIP) and local (RS232) communication.
- Collection and reporting of FGC and power converter diagnostics.
- Logging for post mortem analysis.
- Component identification and temperature measurement using Dallas 1-wire devices.

The first milestone for the project was to deliver 75 FGCs by the end of 1999 to run the converters in the LHC magnet test facility [2][3]. This milestone was met with a first non-radiation tolerant version called the FGC1, shown in figure 1. Seventy-five were produced and were operated in various test facilities from 2000 until 2010.

Figure 2 shows the core architecture of the FGC. It is based on an integer microcontroller (MCU) and a floating point DSP. This architecture was kept unchanged in the evolution from the FGC1 to the final design used in the LHC, the FGC2.

Figure 3: FGC2 developed from 2000-2003.

FGC2 OVERVIEW

At the core of both FGC1 and FGC2 is a Motorola M68HC16Z1 integer microcontroller (MCU) clocked at 16 MHz and a Texas TMS320C32 floating point digital signal processor (DSP) clocked at 32 MHz. Both devices date from the early 1990s and are still in production today. Each processor has its own error-corrected 512KB SRAM. Furthermore, 16KB of the MCU's SRAM is visible to the DSP to enable communication between them. Programs are stored in flash memories connected to the MCU.

Aside from the addition of error detection and correction (EDAC) chips on the SRAM, the biggest change in the FGC2 was mechanical. The circuits were reformatted from four 6U x 160 mm cards to six 6U x 80 mm cards which are mounted on a passive motherboard and enclosed inside a metal cassette, as shown in figure 3.

Glue logic was moved from a Spartan FPGA to eleven smaller flash-based Xilinx 95000 series CPLDs, all mapped into the MCU memory space. Being flash-based these CPLDs do not suffer from corruption of their programming under radiation and they were hoped to be sufficiently tolerant to support the radiation anticipated in the LHC. Unfortunately recent tests have shown that they have a small cross section for latch-up in a high-energy particle field and this will become a problem when the LHC reaches nominal intensity. A replacement design is in development for deployment in 2015 for the systems affected by radiation.

The redundant acquisition of the circuit current is performed by two ADS1201 delta-sigma ADCs. These were the first delta-sigma ADCs from Burr Brown and they do not contain a digital filter. This was implemented externally in Spartan 20 FPGAs which are vulnerable to radiation, so a second analogue interface was developed for the FGCs using MAX337CWI SAR ADCs. The noise and linearity are worse than the ADS1201 but the tunnel-based systems require less precision and these ADCs are sufficiently radiation tolerant to operate next to the LHC.

So far, the FGC2 has been a very successful design for the LHC which is currently operating at about 5% of nominal radiation levels [4]. Its strong points are, in particular:

- Use of the robust WorldFIP fieldbus for both data and synchronisation [5].
- Use of the Dallas 1-wire bus allowing identification of components and measurement of the temperature.
- Metal cassette providing a robust mechanical enclosure.

The FGC project delivered 1750 controllers for operation in the LHC in 2007 at a cost of 7 million Swiss Francs and 50 FTE years. Hardware development took six years and the software took 8 years so there was a strong motivation to re-use as much as possible when developing the next generation FGC.

FGC3 DESIGN

An evolution of the FGC2 was the chosen for the control of new power converters in the PS Booster and the new LINAC 4. The primary objective was to improve performance to allow a higher current regulation bandwidth while reducing costs by re-using as much as possible from the FGC2 development. A key choice was to not maintain the radiation-tolerant specification of the FGC2 since the FGC3 will not be deployed in radiation areas; this has a big impact on the cost of development and production.

Mechanics

The specification for the new controls electronics included the requirement that the crate be only 3U high, compared to 6U for the FGC2. Fortunately the increased density of modern components allowed the FGC3 to fit within a cassette that is half the height and width of the FGC2. The length increased from 160 to 220mm so the volume is reduced by a factor of 3. Figure 4 shows an FGC3 inside its cassette. It can run with passive ventilation and dissipates 8W. The rear connector is a

Figure 4: FGC3 developed from 2007-2011.

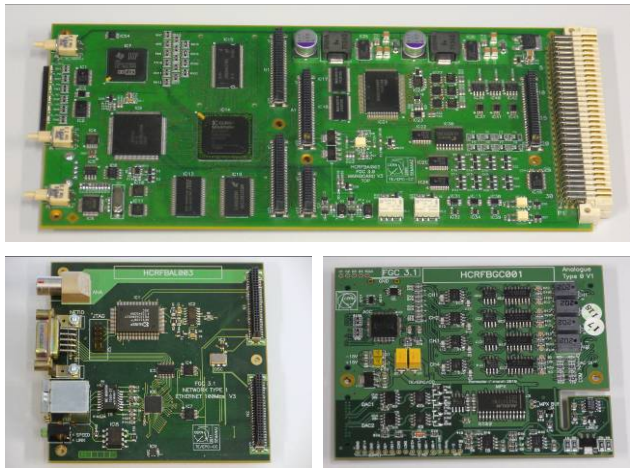


Figure 5: FGC3 motherboard, FGC_Ether and analogue interface daughter boards.

5-row 160-pin DIN connector (as used for VME 64 cards).

The FGC2 structure of a passive motherboard and six daughter boards is replaced by a densely integrated motherboard with two mezzanine daughter boards, as shown in figure 5.

Daughter boards

Anticipating an evolution of the network and analogue interfaces of the FGC3, these functions were moved onto two daughter boards.

When the FGC3 was designed, WorldFIP was the only network choice available, so the first network daughter board provides identical behaviour to the FGC2's WorldFIP interface. Software support for WorldFIP will be maintained indefinitely to allow an FGC3 to operate in the non-radiation areas of the LHC. Alstom stopped developing WorldFIP in 2001 so in 2010 we adopted Ethernet as our future fieldbus (see below). Thus a second variant of the network daughter board now exists (shown in figure 5) and this will be used in all non-LHC installations.

The first use of FGC3s is with small (±10A, ±30V) power converters in the PS Booster. For these an analogue daughter board has been produced with four high precision ADC channels (ADS1274) and two 16-bit DACs (MAX5541). This ADC is a sixth order delta-sigma design with a signal bandwidth up to 50 kHz. We implement our own low-latency digital filters in the FPGA on the motherboard, running at up to 50 ksps. The next application of the FGC3 will be with fast-pulsed converters based on capacitor discharge. These require two higher speed acquisition channels (500 ksps for up to 4 ms) so a second type of analogue daughter board has been designed with two ADS1675 delta-sigma ADCs as well as two slower channels using an ADS1274.

Processors

Modern MCUs or DSPs can easily outperform the combined processing power of the MCU and DSP in the FGC2. However, maintaining the architecture of an MCU and a floating point DSP does not present major disadvantages and allows the re-use of the FGC2 software. The required boost in performance comes from the Renesas RX610 MCU and Texas TMS320C6727b DSP. In the first design in 2007 the little-endian Renesas M32C/87 MCU was selected because of our long experience working with the Renesas M16C and M32C families. This choice required some low-level modifications in the software due to the endian difference with the FGC2's big-endian Motorola M68HC16Z1.

In 2010 the Renesas RX family became available and the Renesas roadmap presented it as the future for the M32C family. As well as having a higher clock rate and a floating point unit, the RX has selectable endianess so we could choose big endian and maintain compatibility with the HC16.

The change in MCU family introduced an eight month delay in the FGC3 project just as we were about to launch pre-series production. But the cost of this short-term delay was considered well worth the benefit given the long-term perspective of the design. More than 1000 units are expected to be produced over the next ten years.

Programmable Logic

The FGC3 logic is concentrated in one Xilinx Spartan 3AN/700 FPGA. Many of the designs were migrated from the FGC2's CPLDs with only minor modifications. This particular FGPA includes an EEPROM to hold its configuration which reduces the chip count by one and is better value than an FPGA plus external EEPROM.

One weakness in the FGC2 design was the inability to reprogram the CPLD logic remotely via the WorldFIP network, since the logic was needed for the processors to run. To avoid this restriction in the FGC3, the MCU and network interface can work without the FPGA. A new logic program can be received via the network and written by the MCU to the EEPROM inside the FPGA.

Synchronisation and communications

Synchronisation is a key requirement for the FGC. Both FGC2 and FGC3 need a 50 Hz sync pulse to discipline a phase-locked loop. In the FGC2 this is a digital PLL based on a fixed quartz oscillator while in the FGC3, clock purity is more important so a VCXO is driven by a 14-bit DAC. Thus the PLL has digital phase detection and a software PI algorithm combined with an analogue oscillator; but it still needs the 50 Hz sync.

When using the WorldFIP network, the sync pulse is simply the IRQ signal from the MicroFIP interface device. This works because WorldFIP is deterministic

Figure 6: Ethernet switch and sync pulse injector.

and the jitter is ±1.5 µs. Ethernet is not deterministic so a separate 50 Hz sync signal is routed to the FGC on a spare pair of wires in the UTP cable carrying the 100 Mbps Ethernet. The jitter on this sync signal is much less than 100 ns. To make this work, a 24-port sync pulse injector box is mounted under each 24-port Ethernet switch and each link is routed via the box. This is shown in figure 6.

The 50 Hz comes from a timing receiver in the gateway PC and is routed on a coax cable running in parallel with the gigabit Ethernet backbone which links the gateway to the switches. The extra cost and space required for this custom designed box is minimal and the reward is considerable: completely standard low-cost Ethernet infrastructure and components can be used. The gateway is a rack-mounted PC with two Ethernet interfaces built-in. The unmanaged switches are cheap commodity items and 100 Mbps LAN chipsets as used in the FGC3 are also widely available from numerous suppliers.

The FGC communication protocols are the same running on WorldFIP packets or raw Ethernet packets. The cycle time remains 50 Hz in both cases. Throughput is far higher with Ethernet because packets are up to 1500 bytes instead of 120 bytes and multiple Ethernet packets can be sent in the same cycle. The gigabit backbone has enough capacity to allow up to 64 FGC3s to be linked to each gateway (instead of 30 over WorldFIP) and for all to communicate at maximum rate simultaneously.

Network address

The network address of an FGC on WorldFIP (1-30) is encoded on a small passive kaptan circuit board soldered into the DB9 connector. Thus FGC cassettes may be changed but the network address (and thus the system's identity to the control system) remains with the connector.

This has proven extremely effective and reliable and to allow a similar solution with Ethernet the bottom byte of the MAC address is coded on a passive circuit board soldered into a DB9 dongle that must be plugged into the front panel of the FGC3 in order to release the system from reset. At boot up, the FGC3 reads the network address (1-64) from the dongle and uses it to form the MAC address that it writes into the SMSC LAN9221 Ethernet interface chip.

SOFTWARE TOOLS

The FGC2 software environment is based on a commercial IDE from RistanCASE GmbH called DAC and two commercial compilers; the HC16 HI-CROSS+ compiler from Hiware and the TMS320C3x/4x C compiler from Texas Instruments. Both were developed for Windows 98 and have been frozen since 1999. So despite being commercial products, neither is supported and compatibility with future versions of Windows is not assured. This is a concern given the anticipated 20 year lifetime for the FGC2.

The FGC3 software environment is based on open source tools. The IDE is Eclipse with Git for version control. The GNU gcc toolchain generates code for the Renesas RX610. It can also generate code for the TMS320C6x family but for the moment we are using a proprietary compiler from Texas.

Some bugs were detected in gcc for the RX and the M32C (when it was being used) but each time the availability of the source code and the large community that supports the gcc toolchain allowed us to fix the bugs and generate a new compiler in less than five days. This experience encourages us to migrate to gcc wherever possible.

We have chosen NewLib as the ANSI C library for the RX because it supports multi-threaded applications and is perfectly suited to small embedded systems.

The real-time OS used on the MCU of both the FGC2 and FGC3 is called NanOS. It is an in-house development derived by simplifying µC/OS-II [6].

CONCLUSIONS

Despite the completely different MCUs, DSPs and fieldbus technologies, the FGC3 is running the software developed for the FGC2. Both programs share a common code base and share about 90% of their source code. By maintaining the core architecture combined with the flexibility given by the programmable logic and the separation of functionality into different modules, both generations of FGC appear almost identical at the conceptual level.

The porting and debugging of the low-level functions to the new FGC3 took about 1 year, while for the higher level functions it took another 6 months. By comparison the development of the current operational FGC2 software took twelve years. By inheriting technology from the FGC2, the project saved an estimated 30 FTE years compared to starting from scratch.

As hardware design tools (PCB and FPGA) become more powerful, it has become clear that it is the software which becomes the hardest and most time consuming part of an embedded converter controller development project.

REFERENCES

[1] J.G. Pett et al, CERN, "A Strategy for Controlling the LHC Magnet Currents", EPAC'96 p. 2317.

[2] Q. King et al, CERN, "Developments in the High Precision Control of Magnet Currents for LHC", PAC'99 p. 3743.

[3] Q. King et al, "The All-Digital approach to LHC power converter current control", ICALEPCS'01, THBT004

[4] Q. King, "Status of the LHC power converter controls", ICALEPCS'09, MOB003

[5] Q. King, "Advanced uses of the WorldFIP fieldbus for diverse communications applications within the LHC power converter control system", ICALEPS'05, WE4B.3-2O

[6] Jean J. Labrosse, "MicroC/OS-II: The Real Time Kernel", 2002, CMP Media LLC.

FAST SCALAR DATA BUFFERING INTERFACE IN LINUX 2.6 KERNEL

A. Homs#, ESRF, Grenoble, France.

Abstract

Key instrumentation devices like counter/timers, analog-to-digital converters and encoders provide scalar data input. Many of them allow fast acquisitions, but do not provide hardware triggering or buffering mechanisms. A Linux 2.4 kernel driver called *Hook* was developed at the ESRF as a generic software-triggered buffering interface. This work presents the portage of the ESRF *Hook* interface to the Linux 2.6 kernel. The interface distinguishes two independent functional groups: trigger event generators and data channels. Devices in the first group create software events, like hardware interrupts generated by timers or external signals. On each event, one or more device channels on the second group are read and stored in kernel buffers. The event generators and data channels to be read are fully configurable before each sequence. Designed for fast acquisitions, the *Hook* implementation is well adapted to multi-CPU systems, where the interrupt latency is notably reduced. On heavily loaded dual-core PCs running standard (non real time) Linux, data can be taken at 1 KHz without losing events. Additional features include full integration into the /sys virtual file-system and hot-plug devices support.

INTRODUCTION

Many scientific experiments carried out at the ESRF beamlines (BL) are based on the *scan* concept, where the evolution in time of one or more magnitudes is measured, typically while scanning the values of one or more parameters. Two modes of scan execution are identified. In "step-by-step mode", scan steps and measurements are performed sequentially (parameter change does not overlap in time with detector integration), starting and stopping the involved hardware at each point. In "continuous mode", both scan and measurement are executed in parallel, increasing the duty cycle of the X-ray beam utilisation, and/or notably reducing the total scan duration. As a drawback, time errors in the synchronisation of the readout of all the involved magnitudes affect the precision of the continuous scan.

Initial support to continuous mode at the ESRF was based on a "soft" synchronisation. In the first versions, a scan loop sequentially read, as fast as possible, motors and counters in a VME crate from a remote control workstation. The next approach read all these scan magnitudes sequentially inside a program running in the VME crate, removing the network from the sources of synchronisation error. A third performance improvement used a combination of hard and soft synchronisations. A hardware event triggered an interrupt request (IRQ), and during its IRQ service routine (ISR) all the scan scalar parameters (implemented as registers in instrumentation boards) were read and stored in a kernel buffer.

This IRQ *Hook* mechanism, originally developed in OS/9, was extended in Linux 2.2 [1] and 2.4 kernels with a flexible configuration interface. A dedicated TACO device server allowed the configuration and reading of the kernel buffer by the BL control program, running on the same PC or on a remote workstation.

In the migration of the VME and PCI instrumentation control at the ESRF to Linux 2.6, the new kernel device abstraction was extensively used [2]. In particular, a robust VME infrastructure was developed for Symmetric-Multi-Processor (SMP) environments, supporting device hot-plug functionality.

HOOK INTERFACE

Device Abstraction

Virtually all the common instrumentation hardware providing scalar parameters can fit in the simple device/channel concept. A hardware unit (VME/PCI board, serial line/GPIB/Ethernet instrument) is considered in this context a device. It groups control registers or commands, and exports one or more scalar channels. In addition, a device can generate events, typically hardware IRQs or operating system (OS) signals. These events are used in the *Hook* structure to trigger the readout of an arbitrary set of channels (belonging to the same and/or other devices), which are recorded in dedicated buffers.

Like its Linux 2.4 predecessor, the *Hook* interface is seen as a "kernel service" that instrumentation drivers register to using their names. This interface has been completely redesigned in Linux 2.6 to explicitly distinguish device, event, and channel structures. The same kernel "device" concept already identified by those drivers by a *major:minor* pair, like a specific PCI board structure associated to a kernel *pci_dev*, can contain an analogue *hook_dev* structure. Each *hook_dev* is the parent of a group of *hook_chan* and/or *hook_event* structures, depending of the board functionality.

Once all *hook_devs*, with their *hook_events* and *hook_chans*, are registered at drivers' start-up, they are ready to be used. The "user" side of the *Hook* "service" configures for an acquisition (*Hook* buffer) the event generator and the data channels to be read at each event. The event setup is performed through the board-specific drivers because of its complex nature. As a result of this setup, the event generator's driver installs the *hook_event* on the specified *Hook* buffer. The data channel setup is performed through the *Hook* driver interface, by using a list of *<drv_name, minor, chan_idx>* triplets that reference the channels to store on the buffer.

#alejandro.homs@esrf.fr

Each *hook_chan* has an associated *hook_chan_functs*; the structure defining the methods that carry out its specific functionality. During buffer channel setup a *prepare* method is called for each selected channel to perform its hardware initialisation. If it succeeds, the board driver module is locked to avoid to be removed from the kernel, and the *hook_chan_functs.lock* method is called to notify the board driver that the channel is in use. On each point the event generator driver calls the *Hook* interface, and the *read* method is called for all the involved *hook_chans* to trigger their readout. In case the channel value is not available immediately (it is called in kernel interrupt context), the *read* method can return a delayed completion request. Two solutions are identified in such a condition: asynchronous response or "read-before-next". In the former, the board driver is responsible of calling an asynchronous buffer write *Hook* function once the value is ready. The latter is requested if the driver does not provide asynchronous events; it can only ensure that the value will ready before the next event. Then, the *read_before_next* method is called to retrieve the previous channel value. Upon acquisition end the *unlock* method is called to release the channel.

A device can have channels of different classes, and hence have different set of methods, which might be the case of a multi-purpose counter/timer, analogue and digital I/O PCI board.

The current *Hook* implementation is Linux 2.6 kernel is limited to VME and PCI boards, which generate hardware IRQs and can be controlled by standard I/O operations. Communication-based devices using serial line, GPIB or Ethernet interfaces are not covered (yet). This limitation is due to both the complexity in accessing these platforms from the kernel interrupt context and the latency normally associated to those instruments. Nevertheless, Linux 2.6 kernels are so flexible and well structured that the support of those interfaces would be, in principle, possible.

Buffer Functionality

As we have seen, a fast acquisition involves an event generator, a set of data channels and a kernel buffer. Multiple acquisitions, each with its own buffer, can be performed in parallel on the same PC; they might share data channels if the *prepare* and *lock* methods' implementations allow it. The *Hook* driver creates 5 buffers by default, but this can be changed at module load time.

Two buffer filling modes are implemented: linear, where the acquisition stops once the buffer is full; and circular, where a ring buffer is implemented and data must be read continuously to not get overwritten. Data overrun is not considered a fatal error, but it is always reported.

In addition to kernel memory allocation and management, each *Hook* buffer is in turn a *hook_dev*. In this sense it exports two *hook_chans*: the point index and elapsed time in microseconds, both measured since the beginning of the acquisition. These two values are very important to determine if an event has been lost, either by the kernel (elapsed time between adjacent points higher than expected) or by the user-space program reading buffers (discontinuity in point index). Each buffer also exports a *hook_event*: a Linux software timer, whose period can be set with a time resolution given by the kernel HZ timer frequency (HZ=1000 in RedHat EL 4: 1 ms resolution).

A simple user-space library allows to configure the kernel interface and to read the buffer contents in an efficient way. As mentioned before, a TACO server on top of the library allows clients like SPEC [3], the central BL control application, to retrieve the data on the buffers. For ring-buffer acquisitions with intervals below 1 ms, the readout must be performed continuously due to the limited length of kernel buffers. For these fast acquisitions, the TACO server provides a bigger, user-space buffer and an auxiliary thread to empty the kernel buffer accordingly. This relaxes the time constrains of the remote SPEC main acquisition loop.

Virtual sysfs Support

The Linux 2.6 */sys* virtual file-system gives a useful user-space visibility of kernel structures. The *class* interface is used in the *Hook* to export information about the registered elements, replicating its tree structure of *hook_drv*, *hook_dev*, *hook_chan* and *hook_event*. In this direction, additional *read_desc* methods in *hook_chan* and *hook_event* are foreseen to provide a human-readable description through */sys* interface.

The *Hook* driver allows code tracing and debugging through messages on the standard kernel *syslog* interface. However, this is far from optimal for very fast acquisitions. In order to provide real time status to user space, the */sys* interface also publishes the instantaneous buffer information. It includes buffer setup (size, filling mode, list of channels) and buffer state (total number of events, number of points to be read and elapsed time since start).

IMPLEMENTATION

Supported Hardware

The standard ESRF BL instrumentation PCI and VME boards used for fast acquisition has been ported to the Linux 2.6 Hook structure:

- ESRF PCI P201 Counter/Timer
- ESRF VME VCT6 Counter/Timer
- Compcontrol CC133 Incremental Encoder
- ADAS ICV150 Analog-to-Digital Converter (ADC)

The first two boards implement both data channels and event generators; the last two only export data channels.

SMP and VME Issues

Linux 2.6 gives an improved support to hot-plug dynamics, that is, devices that can appear and disappear without system control. This is the case of VME devices connected to PCI/VME bus couplers; if the remote crate is disconnected the corresponding VME devices no longer exist on the system. It becomes an important issue in SMP

systems, because one or more CPU/cores could be referring to one device while another is trying to delete it. Such situation is even more critical in the *Hook* environment, which works on interrupt context and the system can immediately hang on invalid operations. Special care has been taken to protect all the data structures against these different situations.

Nevertheless, some crashes or conflicts are unavoidable if performance must be kept as first design priority. One example is found on scans with a PCI IRQ-based event generator that reads channels in VME boards. The conflict arises if the remote VME bus access (channel *read)* is done while another CPU/core is performing an Interrupt-Acknowledge cycle on the same VME bus. This condition is forbidden by the PCI/VME bus coupler, and implementing additional protection mechanisms would sacrifice the performance of fast acquisitions.

PERFORMANCE

Measurement Details

The current ESRF standard instrumentation control computer is based on 4U industrial PCs with Intel Core 2 Duo E6400 @ 2.13 GHz. The OS is RedHat EL 4 [Update 6]; the official Linux kernel 2.6.9-67.ELsmp has not been modified. Measurements have been performed on that system with two PCI counter/timer boards, and one SBS Bit-3 PCI/VME bus coupler with all the supported VME boards mentioned before. The event generator is either a PCI and/or VME timer programmed in free-run (cyclic counting), with end-of-count (EOC) IRQ activated. Another counter of the same board is set to count microseconds, and to latch its value on timer EOC. By reading the running counter value as first data channel and then its EOC-latched value, it is possible to calculate the effective IRQ latency. Moreover, by specifying the running counter value twice consecutively in the channel list, the channel access time is obtained, quantifying the systematic component of the synchronisation error.

Single Acquisitions

For the first tests the VCT6 channel 1 was configured as free-run timer for event generation and channel 2 as a microseconds counter, as explained before. The *Hook* buffer was setup to store 10 channels:

- VCT6 channel 2 running counter (read 4 times)
- VCT6 channel 2 latch counter
- *Hook* buffer 0 point index
- *Hook* buffer 0 time stamp (read twice)
- ICV150 channels 1 & 2

The measurement results for a timer event frequency of 5 KHz are shown in Table 1. Trying to measure at higher event frequencies resulted in data point lost.

A common conclusion for all the measured magnitudes is their small standard deviation (SD), showing that, in most of the time, the Linux SMP kernel is very regular in scheduling. The notable difference in VCT6 and P201 average IRQ latencies comes from the delay added by the PCI/VME bus coupler hardware and software layers.

Table 1: VCT6 IRQ Latency and Read Delays (µs)

Magnitude	Min.	Ave.	Max.	SD
VCT6 IRQ latency	15	24	126	0.8
VCT6 channel read delay	0	2.5	150	0.8
Hook channel read delay	2	3.1	22	0.7

The results for the same measurements using the P201 are shown in Table 2. Again, 5 KHz is the maximum readout frequency for stable acquisitions.

Table 2: P201 IRQ Latency and Read Delays (µs)

Magnitude	Min.	Ave.	Max.	SD
P201 IRQ latency	3	4.2	105	0.6
P201 channel read delay	0	0.9	7	0.25
Hook channel read delay	2	3.1	26	0.7

When analysing the maximum IRQ latency values, it should be noted that these conditions are close to the data sampling reliability limits. As it will be seen in the next section, lower sample frequencies give smaller differences between min./max. values. Indeed, the higher sample frequency, the larger amount of data per unit time must be transferred from kernel to user-space, and then to the TACO client via the Ethernet device, reducing the total idle times of CPUs, chipset and hardware busses.

In this sense, IRQs shared with active devices like network and disk controllers are a proven source of instabilities. The limited number of 4 available PCI IRQs in standard systems makes sharing difficult to avoid. It is better then to choose inactive devices like embedded USB or graphics controllers, unused in most of our applications.

It should be noted that jitter in ISR scheduling is important if the channels to be read are not hard synchronised with the event generator. This is the case of many encoders and ADCs. If, on the contrary, the boards allow data latching on hardware events, like VCT6 and P201 counters, this jitter does not contribute to the measurement error, and it is only important to guarantee that no point is lost by the kernel.

Parallel Acquisitions

It was explained before that the *Hook* has been designed to support parallel acquisitions. This can be useful when a monitoring process turns in the "background" during fast scans, or when devices with different speed capabilities are brought together on the same measurement.

We have performed both uncorrelated and correlated parallel acquisitions with the VCT6 and the P201 reading VME and PCI channels, respectively. In the uncorrelated case, both timers generate IRQs at the same frequency but

with unlinked phase, creating both coincidence and anti-coincidence conditions during very long scans due to the independence of their internal oscillators. In the second case, the VCT6 is the master timer and two P201 boards have been chained as slaves, so the VME and PCI IRQs synchronisation is at a sub-microsecond level.

Unlike previous measurements, where the acquisition was the only running task in the system, the parallel uncorrelated scans have been performed under heavy system load. Besides the fast acquisition, the system was executing:

- RedHat kernel recompilation using rpmbuild
- Intensive network data transfer with *iperf*-like program
- Linux root (/) partition imaging in a file on a user partition using basic OS tools.

Each of these tasks was continuously executed in *bash* with the basic loop *while true; do …; done*. The OS load ranged from 2.5 – 4.5, with an average of 3.5. The uncorrelated statistics under these conditions are shown in Table 3.

Table 3: IRQ Latencies (µs) in Uncorrelated Parallel Scans @ 1 KHz on Heavily Loaded System

Magnitude	Min.	Ave.	Max.	SD
VCT6 IRQ latency	14	26	80	3.0
P201 IRQ latency	3	6	110	8.5

Like in the previous cases, the tests ran for 24 hours, and no scan point is missing. Again, no specific tuning, or kernel patching has been applied; the default RedHat configuration with the *CONFIG_PREEMPT=no* kernel option has been kept.

The correlated measurements were performed on an unloaded system to double the scan rate. Table 4 shows the corresponding results.

Table 4: IRQ Latencies (µs) in Correlated Parallel Scans @ 2 KHz on Unloaded System (2x P201)

Magnitude	Min.	Ave.	Max.	SD
VCT6 IRQ latency	14	25	70	1.6
2nd P201 IRQ latency	45	50	104	2.0

The shift in the P201 IRQ latency comes from the fact that it is measured on the second board, on a different PCI slot that shares IRQ with a larger number of devices.

DEPLOYMENT AT THE ESRF

The Linux 2.6 *Hook* interface has been in production on ID31 Powder Diffraction ESRF BL for more than one year. It is the core of the BL experiments, which are normally carried out between 60 – 300 Hz sample rates (3 – 15 ms). System instabilities have been completely fixed during this time, including vulnerabilities against crashes of the remote BL control workstation. It is planned to be deployed to the other of ESRF BLs that use fast scans during the next long shutdown, in the framework of a global SuSE 7.2 → RedHat EL 4 migration campaign.

CONCLUSIONS

A generic kernel *Hook* interface, providing fast software trigger and scalar data buffering, has been ported to Linux 2.6. Good performance results are obtained with standard (non real time) RedHat EL 4 kernels running on mid-level dual-core CPUs. Up to two parallel, uncorrelated acquisitions can be performed at 1 KHz on heavily loaded systems without data lost. Support to */sys* virtual file-system provides useful information for *Hook* configuration, monitoring and debugging. The new module has been in production at the ESRF for more than one year; its deployment to the rest of ESRF BLs is foreseen soon.

ACKNOWLEDGEMENTS

The author wants to thank F. Sever, M. Perez, E. Papillon, G. Berruyer, J.M. Clement and H. Gonzalez from the Instrumentation Services and Development Division (ISDD) at the ESRF for their support, help and useful discussions during the development of this work.

REFERENCES

[1] A. Homs-Purón, D. Beltrán, A. Beteva, M. C. Domínguez, P. Fajardo, A. Götz, J. Klora, E. Papillon, M. Pérez, V. Rey, "Linux/PCI: The ESRF Beamline Control System Modernisation", ICALEPCS'03, Gyeongju, October 2003, MP565, p. 162 (2003), http://www.JACoW.org.

[2] A. Homs, F. Sever, "Generic VME Interface for Linux 2.6 Kernels", PCaPAC'08, Ljubljana, October 2008, TUP001, p. 77 (2008), http://www.JACoW.org.

[3] SPEC - X-Ray Diffraction and Data Acquisition Software, G. Swislow, Certified Scientific Software, http://www.certif.com

DEVELOPMENT OF IMAGE DATA ACQUISITION SYSTEM FOR 2D DETECTOR AT SACLA

A. Kiyomichi*, A. Amselem, T. Hirono, T. Ohata, R. Tanaka, M.Yamaga,
JASRI/SPring-8, Sayo, Hyogo, Japan
T. Hatsui#, RIKEN/SPring-8, Sayo, Hyogo, Japan.

Abstract

We have developed a data acquisition system for X-ray and optical imaging devices at SACLA (SPring-8 Angstrom Compact free electron LAser). The frontends with the Camera Link interface transfer the data to data-handling servers, which buffer and distribute the data to on-line monitors and a high-speed storage. The system is partly operational for SACLA beam commissioning since March 2011. The full system for X-ray 2D detector experiments will be released in November 2011.

INTRODUCTION

The X-ray free electron laser (XFEL) facility, SACLA, has been constructed at the SPring-8 site. The XFEL has unprecedented characteristics such as high peak brightness, ultra-short pulses, and full spatial coherence. These features are believed to open new opportunities in a variety of scientific fields. The facility has started to accelerate electron beams up to 8 GeV in February 2011, and the first X-ray laser was observed in June 2011.

In order to meet the requirements of proposed experiments to be performed at SACLA, Multi-Port Charge-Coupled Device (MPCCD) detector has been developed. The MPCCD sensor has 0.5 M pixels with 50 um square pixels. It is packaged in a 4-side buttable arrangement in order to enable large area detector by forming tiled array [1]. In the first experimental runs starting from March 2012, two arrayed detectors with 8 and 4 sensors in tandem will be the operation with maximum data rate reaching 5.8 Gbps. Each sensor is driven by a readout board, which transmits the digitized data through a Camera Link base configuration interface. The resulting data is processed on the VME Camera Link board, and transferred to a data-handling server via TCP/IP socket communication on 1 Gbps Ethernet. The data are pipeline buffered on the server, and sent to the on-line data visualization terminals, and a data file server connected to the high-speed data storage. The data acquisition is carried out completely in parallel so to ensure the scalability for detectors with arrayed sensors.

FRONTEND

We have developed the image data-handling scheme using the event-synchronized data-acquisition system [2]. We selected the Camera Link standard to achieve real-time triggering and high-speed data transfer. We have made available a VME Camera Link board for base configuration cameras in addition to the off-the-shelf PCI-express Camera Link board (AVALDATA APX-3318) for medium and full configuration devices. The former board has been developed in order to achieve on-the-fly lossless compression.

VME Camera Link Board

The VME Camera Link board is composed of a Camera Link base configuration interface and a processor PMC (PrPMC) mezzanine cards onto a carrier VME board with a FPGA [3]. Figure 1 shows VME Camera Link board. The data is first acquired via the Camera Link interface card and sent to the FPGA, Virtex-5 FXT (XC5VFX70T), and buffered to a DDR2 memory. Buffered data are compressed by a lossless range-coder algorithm within the FPGA, and transmitted to the buffer in the SDRAM on the PrPMC card. On the PrPMC card, TCP/IP socket server process is running on Freescale Semiconductor embedded Linux. A client process on the data-handling server receives the data through a Gigabit Ethernet interface. The achieved overall data rate in this scheme was 760 Mbps, which outperforms the required bandwidth of 480 Mbps. The carrier board has another Gigabit Ethernet interface. The Linux hard core on the FPGA has another TCP/IP socket server process on the TimeSys Linux, which can deliver the uncompressed data at the reduced frame rate for on-line monitoring. The measured data rate was 200-300 Mbps. Currently we have implemented MPCCD detector and a VGA camera for the screen monitors of the beamline.

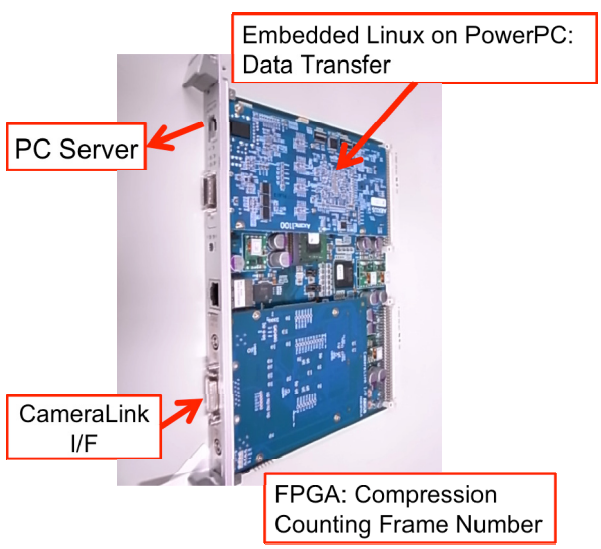


Figure 1: VME Camera Link board.

* kiyomichi@spring8.or.jp
# hatsui@spring8.or.jp

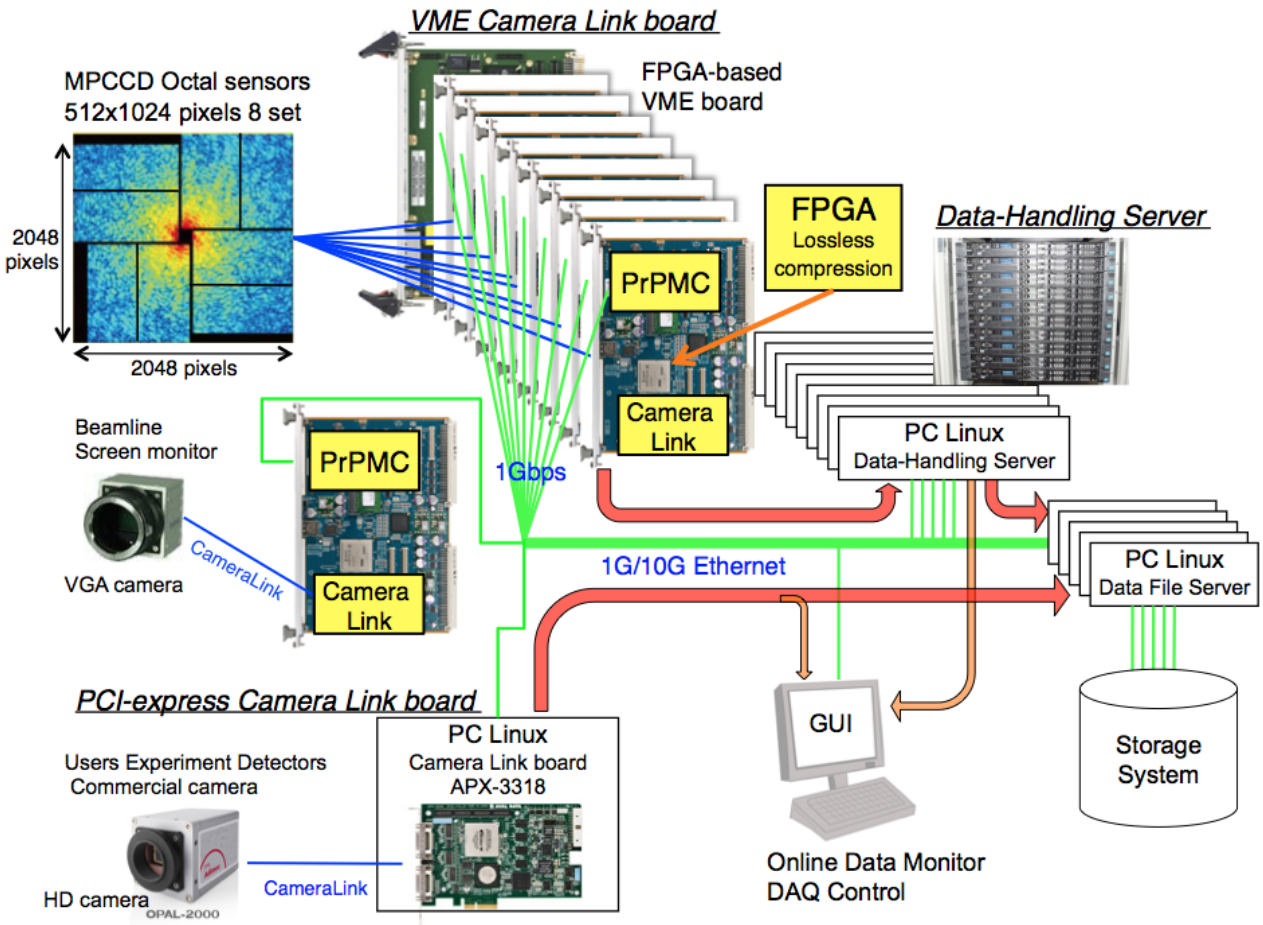


Figure 2: Schematic view and data flow of the image data-acquisition system (VME and PC).

Off-the-Shelf PCI-Express Camera Link Board

The PCI express Camera Link board (AVALDATA APX-3318) is employed on the operating system CentOS 5.4 for the sensor calibrations of MPCCD detectors. Long-term stability of the data transfer performance with this board in conjunction with high-definition (1920x1080 pixel) CCD cameras (Adimec OPAL 2000) has been successfully examined under the SACLA DAQ system. The stability verification with scientific CMOS camera (pco.edge PCO AG, 2560x2160 pixel) is now in progress. The current status of the VME-based and the PC-based Camera Link data acquisition are summarized in Table 1.

DATA-HANDLING SERVER

Figure 2 shows the schematic view and data flow of the VME-based and the PC-based image data acquisition systems. The data-handling server receives the data from the VME Camera Link board by TCP/IP socket connection. Each sensor data is transferred to the corresponding data-handling server, and stored temporally to a ring buffer in the server memory after formatting the data structure. Another process on the data-handling server reads the data from the ring buffer and provides a TCP/IP data transport service. A client process on a data file server receives the data and writes them to the connected high speed storage comprising of StoreNext single file system on the DDN (Data Direct Network) S2A9900 Storage. Total data throughput with 12 MPCCD sensors reaches 5.8 Gbps was achieved without any loss of the data.

Table 1: Current Status of VME-based and PC-based Camera Link Data Acquisition

	VME system	PC system
No. of support cameras	2	>100
Supported Camera Link configuration	base only	Base, medium full
Camera clock	< 80MHz	< 85MHz
Loss-less compression	Yes	No
MADOCA control	Yes	Yes

Another server process on the data-handling server offers data transmission service to the on-line data monitors. The process can provide basic analysis features such as averaging, region-of-interest statistical analysis, line profile calculation etc. User can also implement their own analysis code on the data-handling server to monitor

their experimental conditions. All these client-server processes as well as the board configuration control processes are controlled by the MADOCA framework [4] in order to achieve rapid implementation in the distributed SACLA DAQ system.

BEAM COMMISSIONING

The electron beam commissioning at SACLA started in February 2011, and the first light was observed in March. After three months of beam tuning, the first X-ray laser was observed in June. In the course of the commissioning, the DAQ system has been employed extensively for X-monitoring the spontaneous X-ray from the undulators, and X-ray laser characteristics.

SUMMARY

The image data acquisition system has been developed for the X-ray 2D detector and optical cameras at SACLA. We have developed a VME Camera Link board. We demonstrated that the data transfer from the MPCCD detectors to the high-speed data storage at a transfer rate of 5.8 Gbps. The off-the-shelf PCI-express Camera Link board was also implemented in the system in order to support wider range of cameras.

ACKNOWLEDGEMENT

The authors would like to thank Mr. Yu Kirino, Mr. M. Nagasaki, Mr. Y. Yoshioka and other members of ARKUS Corporation for the development of VME Camera Link board.

REFERENCES

[1] T. Kameshima, et al., “Development status of X-ray 2D Detectors for SPring-8 XFEL”, IEEE Nuclear Science symposium N13-4 (2010)

[2] M. Yamaga, et al., “Event-Synchronized Data Acquisition System of 5 Giga-bps Data Rate for User Experiment at the XFEL Facility, SACLA” in these Proceedings of ICALEPCS2011, Grenoble, France

[3] T. Hirono, et al., “Application and Upgrade of Flexible and Logic-Reconfigurable VME Board” in Proceedings of ICALEPCS2009, Kobe, Japan, (2009) 221

[4] R. Tanaka, et al., “The first operation of control system at the SPring-8 storage ring”, in Proceedings of ICALEPCS97, Beijing, China, 1 (1997).

POWER SUPPLY CONTROL INTERFACE FOR THE TAIWAN PHOTON SOURCE

C.Y. Wu, Jenny Chen, Y.S. Cheng, P.C. Chiu, Demi Lee, C.Y. Liao, K.B. Liu, K.T. Hsu, K.H. Hu, C.H. Kuo,
NSRRC, Hsinchu 30076, Taiwan

Abstract

The Taiwan Photon Source (TPS) is a latest generation synchrotron light source. Stringent power supply specification should be met to achieve design goals of the TPS. High precision power supply is equipped with 20, 18, and 16 bits DAC for the storage ring dipole, quadrupole, and sextupole magnets with Ethernet interface. Control interface include basic functionality and some advanced features which are useful for performance monitoring and transient diagnostics. These power supplies can be access by EPICS IOCs. Corrector power supplies control interface is a special design embedded interface module which will mount at the corrector power supply crate to achieve required performance. The setting reference of the corrector power supply is generated by 20 bits DAC and reading is done by 24 bits ADC. The interface module is embedded with EPICS IOC for slow control. Fast setting ports are also supported by the internal FPGA for orbit feedback.

INTRODUCTION

The TPS is a latest generation of high brightness synchrotron light source which has been under construction at the National Synchrotron Radiation Research Center (NSRRC) in Taiwan since 2010 [1]. It consists of a 150 MeV electron Linac, a 3 GeV booster synchrotron, and a 3 GeV storage ring. The magnets of TPS are important components around strong ring, booster ring and transport lines. These magnets are used to bend electron and keep electron orbit of storing ring and booster ring. In order to achieve these requirements, many types of power supplies will be used for different magnets. The TPS power supplies control interfaces include Ethernet and CPSC (Ethernet). The functionalities, operation interface and preliminary test of power supplies will be shown in this report.

The EPICS (Experimental Physics and Industrial Control System) is a set of open source software tools, libraries and applications developed collaboratively and used to create distributed soft real-time control systems for scientific instruments such as the particle accelerators [2]. Many resources and supports are available as well as numerous applications for accelerator have been developed.

As a result, the EPICS framework was also selected as control system infrastructure for the TPS project. The EPICS platform has been gradually built and tested to control and monitor the subsystems of TPS. Utilizing EPICS channel access mechanism with the specific toolkits, the data can be accessed between the IOCs and the clients.

STORAGE RING POWER SUPPLY CONTROL INTERFACE

TPS power supplies control interface are divided into four categories rather than a unified solution. These four kinds of power supplies will be provided by three different vendors. The reason of this choice is to meet the practical situation from manpower, budget and available vendors.

Table 1: TPS Storage Ring Power Supply Summary

Magnet	Type	Max Current	Stability ppm	Number of PS	Vendor	Control Interface
Dipole	Unipolar	750 A	±10	1	IE Power (Eaton)	Ethernet
Quadrupole	Unipolar	250 A	±10	240	Chroma ATE Inc.	Ethernet
Sextupole	Unipolar	250 A	±50	168	Chroma ATE Inc.	Ethernet
Corrector	Bipolar	± 10 A	±10	HC: 168 VC: 168 Fast HC: 96 Fast VC: 96	ITRI	CPSC Ethernet – EPICS CA SFP – Orbit Feedback
Skew Quadrupole	Bipolar	± 10 A	±10	96	ITRI	CPSC (Ethernet)

ITRI: Industrial Technology Research Institute of Taiwan, R.O.C.

Dipole Magnet Power Supply

The storage ring dipole DC power supply will be equipped with Ethernet interface. Control resolution will be 18 bits effective number, noise and drift will be better than 10 ppm and it had contracted to IE Power [3] (acquired by Eaton Corporation in 2011).

Quadrupole and Sextupole Magnet Power Supply

The intermediate power supply for storage ring quadrupole and sextupole magnet with current rating 250 Amp will be equipped with Ethernet interface as well. The quadrupole magnet power supply is 18bit with higher stability than sextupole magnet with 16 bits. The two kinds of power supplies are both contracted to a local company Chroma ATE Inc. [4] and would have internal data buffer for transient recording capability.

Corrector Power Supply and Skew Quadrupole Power Supply

The small power supply for corrector magnets in the range of ±10 Amp categories will be interfaced to analogue interface directly. The power supply is NSRRC home made and manufactured by Industrial Technology Research Institute of Taiwan, R.O.C. [5]. The controller

interface CPSC (Corrector Power Supply Controller) is dedicated to be designed for both EPICS control system and fast orbit feedback application. Table 1 summarizes the specifications of storage ring power supplies.

Control system for the TPS is based upon EPICS toolkit framework. The EPICS toolkit provides standard tools for display creation, archiving, alarm handling and etc. These toolkits which have various functionalities will be employed to monitor and to control accelerator system.

There are 24 of dedicated cPCI EPICS IOCs are built individually at the 24 cells of storage ring to operate the power supplies respectively. Each IOC is used to control 10 quadrupole magnet power supplies and 7 sextupole magnet power supplies, and one of IOC is appended to control only one dipole magnet power supply.

The cPCI EPICS IOC equipped with the latest generation CPU board will be standardized as ADLINK cPCI-6510 CPU module [6]. The 6U cPCI platform was chosen for the EPICS IOC platform. Local company manufactured crate and CPU module which could provide an economic solution is the major reason. StreamDevice is a feasible tool to build connection between EPICS IOC and power supplies.

The small power supplies applied for both slow/fast correctors and skew quadrupole magnets. There will be 8 power supply modules in one crate and the first slot will be plugged in one CPSC. Besides general control, monitor and configuration, the fast correctors will be also applied for the fast orbit feedback. The CPSC with EPICS IOC is therefore dedicatedly designed and the embedded FPGA will handle fast setting application. Synchronization mechanism and built-in waveform are also supported. Figure 1 shows the overall control of power supplies for each cell of the storage ring.

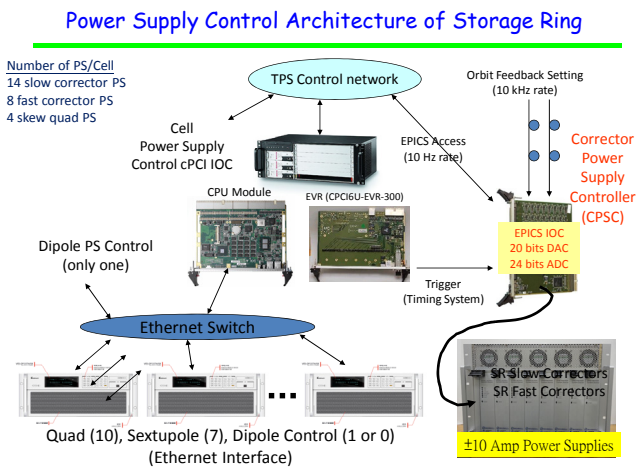


Figure 1: Control infrastructure in one cell of TPS storage ring power supplies.

BOOSTER RING POWER SUPPLY CONTROL INTERFACE

The booster power supplies are composed of one dipole power supply with maximum current 1200 Ampere and four family quadrupole power supplies with maximum current of 130 Ampere. These power supplies will be equipped with the same controller with serial control interface internally. Serial to Ethernet adapter will interface with control system. These power supplies will support external reference input or internal waveform generator for booster power supply ramping. The overall booster ring power supplies control interface is shown in Fig. 2. The most probably booster ramping waveform is sinusoidal wave. Two family of sextupole are driven by two small power supplies. There are 60 horizontal corrector and 36 vertical correctors. Corrector power supplies of the booster will adopt CPSC also. If corrector ramping is necessary, the CPSC has built-in waveform generator which can fulfil this functionality. Table 2 summarizes the specifications of booster ring power supplies.

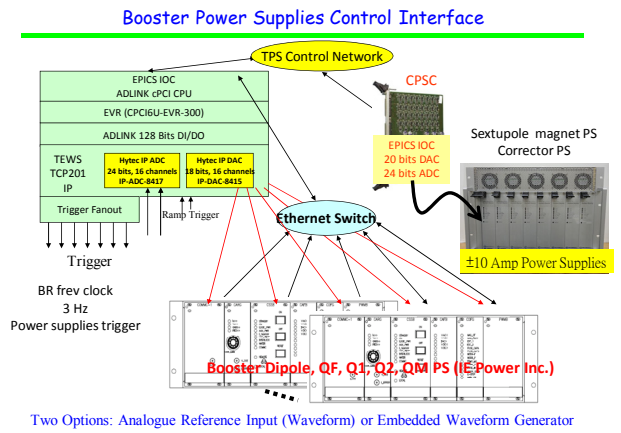


Figure 2: Control infrastructure of TPS booster ring power supplies.

Table 2: TPS booster ring power supply summary

Magnet	Type	Max Current	Stability ppm	Number of PS	Vendor	Control Interface
Dipole	Unipolar	1200 A	±10	1	IE Power (Eaton)	Ethernet*
Quadrupole	Unipolar	130 A	±10	4	IE Power (Eaton)	Ethernet*
Sextupole	Bipolar	± 10 A	±10	2	ITRI	CPSC (Ethernet) with Waveform support in CPSC
Corrector	Bipolar	± 10 A	±10	HC: 60 VC: 36	ITRI	CPSC (Ethernet) with Waveform support in CPSC

TRANSPORT LINES POWER SUPPLY CONTROL INTERFACE

Power supplies for the transport lines includes linac to the booster transport line (LTB) and booster to the storage ring (BTS) dipole magnets, quadrupole magnets and correctors, and DC septa are summarized in Table 3 and Table 4. It was decided that the transport line dipole and quadrupole power supply will adopt the same type power supply as the sextupole power supply of the storage ring to minimize the types of power supply. The rating of the sextupole power supply is 250 Ampere that need parallel two and three sets for the BTS dipole (484 Amp) and DC septa (608 Amp) applications. The power supplies have

the same Ethernet interface with the storage ring sextupole. Despite the corrector for the transport line is much relaxed in the noise specification, the power supply same as the storage ring corrector will be used for this applications to simplify the types of power supply. The corrector power supply for transport line will be controlled by the CPSC module also.

There are two DC septa serve for the booster extraction and storage ring injection to relax the single long pulse septum required due to limited space available for these pulse septa.

Table 3: TPS LTB power supply summary

Magnet	Type	Max Current	Stability ppm	Number of PS	Vendor	Control Interface
Bending	Unipolar	250 A	±10	1	Chroma ATE Inc.	Ethernet
Quadrupole	Unipolar	250 A	±10	10	Chroma ATE Inc.	Ethernet
Corrector	Bipolar	± 10 A	±10	HC: 5 VC: 7	ITRI	CPSC (Ethernet)

Table 4: TPS BTS power supply summary

Magnet	Type	Max Current	Stability ppm	Number of PS	Vendor	Control Interface
Bending	Unipolar	500 A	±50	2 (parallel)	Chroma ATE Inc.	Ethernet
Quadrupole	Unipolar	250 A	±50	7	Chroma ATE Inc.	Ethernet
Corrector	Bipolar	± 10 A	±10	HC: 6 VC: 6	ITRI	CPSC (Ethernet)
BR extraction DC Septum	Unipolar	750 A	±10	3 (parallel)	Chroma ATE Inc.	Ethernet
SR extraction DC Septum	Unipolar	750 A	±50	3 (parallel)	Chroma ATE Inc.	Ethernet

POWER SUPPLY CONTROL ENVIRONMENT

To control and monitor power supplies based on EPICS environment via Ethernet, the clients should be installed the specific EPICS base and the graphical OPI (Operation Interface) toolkits, such as EDM (Extensible Display Manager) and MATLAB (channel access via the labCA module) for EPICS channel access.

STORAGE RING POWER SUPPLY OPERATION INTERFACE

The EDM toolkit was chosen to develop the operation interface. The client console can use the specific EDM page to access the data via PV channel access. The preliminary GUI page of storage ring dipole, quadrupole and sextupole magnet power supplies controls should be shown in Fig. 3. Figure 4 shows the GUI page of storage ring vertical corrector, horizontal corrector and skew quadrupole magnet power supplies controls. The macro name method was regularly used to switch each display page. The main control page was shown critical information for observing status easily, and the main operation process functions are also executed from the panel.

The MATLAB toolkit with labCA is adopted to develop the high level application program for commissioning and diverse operation procedures. The application includes the specific overall power on/off control, degauss process, checking power supply status, operation performance analysis, operation statistics and etc. The various operation processes will be developed and tested according to the various operation modes. The detail control page of power supplies with the trend plot for observing is shown in Fig. 5. It shows the current variation during the degauss process executed. The degauss application is also developed with the specific function of batch process to reduce the peak of power consumption for saving energy.

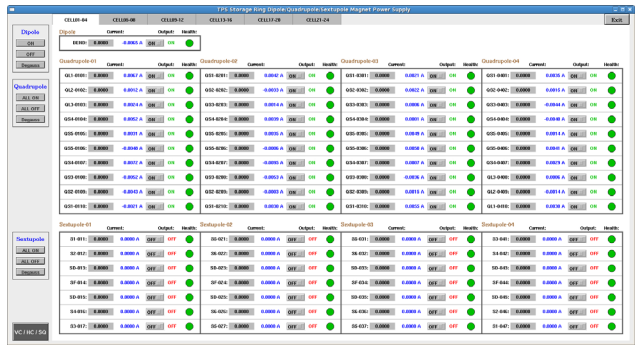


Figure 3: The operation interface of storage ring dipole, quadrupole and sextupole magnet power supplies.

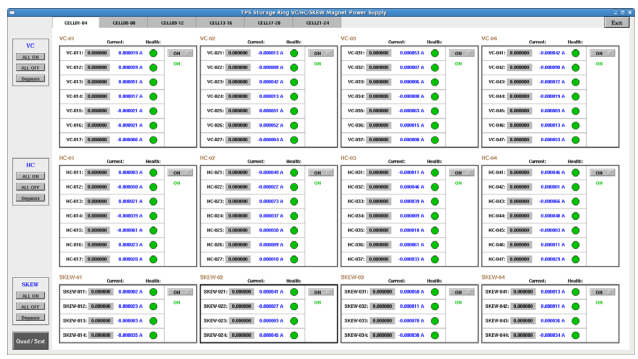


Figure 4: The operation interface of storage ring vertical corrector, horizontal corrector and skew quadrupole magnet power supplies.

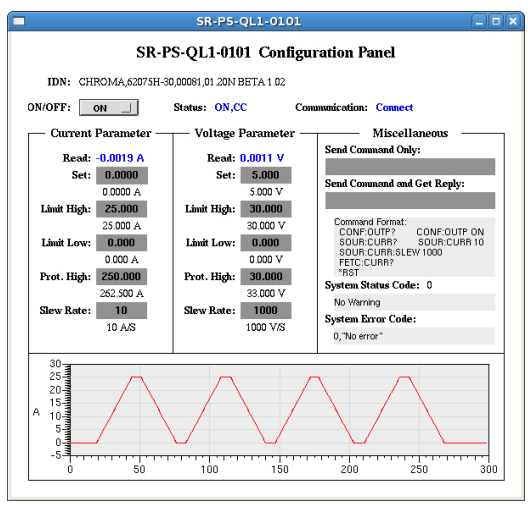


Figure 5: The trend shows current variation of the quadrupole magnet power supply during the simulated degauss process.

PRELIMINARY TEST OF INTERMEDIATE POWER SUPPLY

The several prototype intermediate power supplies for storage ring quadrupole magnet and sextupole magnet have been delivered in NSRRC. These power supplies including 18 bit resolution is used to control quadrupole magnet and 16 bit resolution for sextupole magnet. These power supplies have internal data buffer for transient waveform recorder. The maximum recorder time is 10 seconds with 1k samples/sec and 10000 recorder points. The transient waveform recorder is helpful to analyse behaviours of power supply.

The quadrupole magnet power supply provides sinusoidal wave generator in output DC current of power supply. The frequency and amplitude of sinusoidal wave are adjustable. The sinusoidal wave generator will be used to diagnose power supply and beam physics applications. Figures 6-7 show functionality tests include transient waveform recorder and sinusoidal wave generator for sextupole magnet and quadrupole power supplies. Figure 8 displays current setting resolution test of the prototype sextupole magnet power supply. The resolution is 10 mA with magnet load. The current setting resolution is 5 mA with magnet load for quadrupole magnet power supply is shown in Fig. 9.

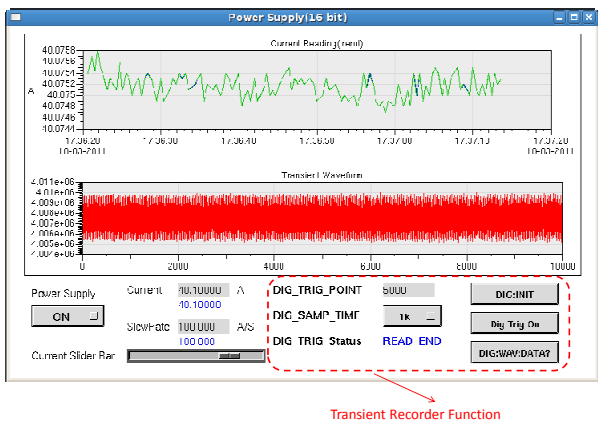


Figure 6: The engineer test EDM screen for functionality of the sextupole magnet power supply.

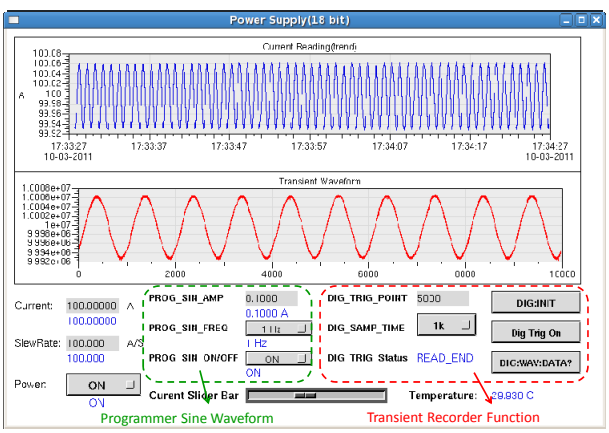


Figure 7: The engineer test EDM screen for functionality of the quadrupole magnet power supply.

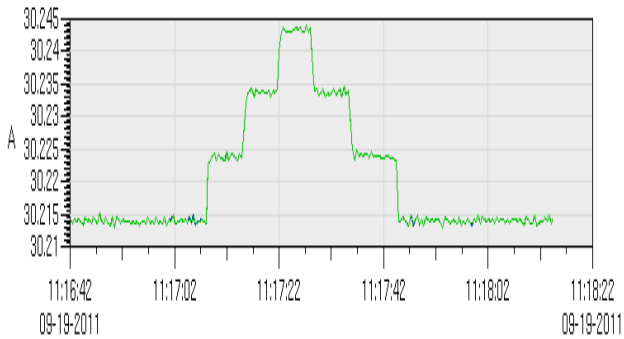


Figure 8: The current setting resolution test of the sextupole magnet power supply.

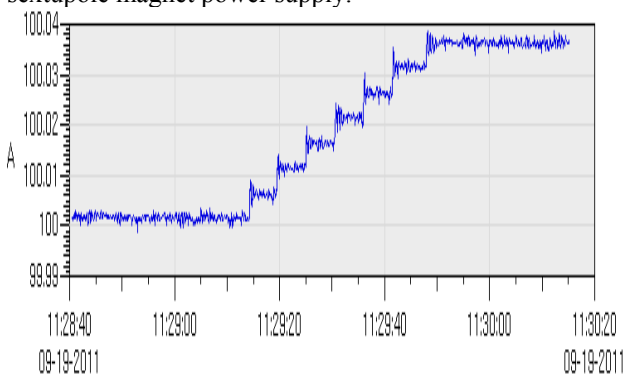


Figure 9: The current setting resolution test of the quadrupole magnet power supply.

SUMMARY

The power supply system of TPS is still in the acquisition phase. Most of power supplies will arrive in 2012. Preparation of the power supply control interface is on going. Prototyping of EPICS support for intermediate power supply of the storage ring, booster ring and transport lines are in proceeding. Plan of the power supply control for main power supply of the storage ring and booster ring is on the way. The CPSC module was contracted to the vendor, detailed design is in proceeding.

Preliminary user interfaces and operation procedures are presented in this report. Before power supplies delivered, the virtual power supply control environment was constantly established for developing the operation applications in advance. The operation applications include the operation interface, power on/off setting and checking, degauss process and etc. The various operation processes will be developed and tested according to the various operation modes. The performance and high level applications of power supplies of TPS will be demonstrated other reports in the future.

REFERENCES

[1] http://www.nsrrc.org.tw/english/tps.aspx.
[2] http://www.aps.anl.gov/epics/.
[3] http://www.iepower.com.
[4] http://www.chromaate.com.
[5] http://www.itri.org.tw/eng/.
[6] http://www.adlinktech.com.

DEVELOPMENT OF PATTERN AWARE UNIT (PAU) FOR THE LCLS BEAM BASED FAST FEEDBACK SYSTEM*

K. H. Kim#, S. Allison, D. Fairley, T. M. Himel, P. Krejcik, D. Rogind, E. Williams, SLAC National Accelerator Laboratory, Menlo Park, CA 94025, U.S.A.

Abstract

LCLS is now successfully operating at its design beam repetition rate of 120 Hz, but in order to ensure stable beam operation at this high rate we have developed a new timing pattern aware EPICS controller for beam line actuators. Actuators that are capable of responding at 120 Hz are controlled by the new Pattern Aware Unit (PAU) as part of the beam-based feedback system [1]. The beam at the LCLS is synchronized to the 60 Hz AC power line phase and is subject to electrical noise which differs according to which of the six possible AC phases is chosen from the 3-phase site power line. Beam operation at 120 Hz interleaves two of these 60 Hz phases and the feedback must be able to apply independent corrections to the beam pulse according to which of the 60 Hz timing patterns the pulse is synchronized to. The PAU works together with the LCLS Event Timing system which broadcasts a timing pattern that uniquely identifies each pulse when it is measured and allows the feedback correction to be applied to subsequent pulses belonging to the same timing pattern, or time slot, as it is referred to at SLAC. At 120 Hz operation this effectively provides us with two independent, but interleaved feedback loops. Other beam programs at the SLAC facility such as LCLS-II and FACET will be pulsed on other time slots and the PAUs in those systems will respond to their appropriate timing patterns. This paper describes the details of the PAU development: real-time requirements and achievement, scalability, and consistency. The operational results will also be described.

INTRODUCTION

SLAC timing system generates 360 Hz fiducial which conducts timing events and is synchronized with the 60 Hz AC power line phase[2, 3]. Thus, there are 6 time slots, each of them corresponding to a different AC power line phase. According to the 120 Hz beam repetition, we need to choose 2 different time slots – TS1 and TS4 for the LCLS, then the LCLS beam runs on the two different AC phases. The different AC phase makes different gain or different response on the actuators – magnets and RF power in accelerator. For the stable 120 Hz beam operation on the LCLS, we need to compensate the different response on the actuators. There are other power line variation sources at the SLAC such as the FACET and LCLS-II which are other beam programs, those are pulsed on other time slots and makes different power line variations. Thus, each of the beam programs makes different power line variations, and needs to compensate for these variations for the stable operation. All of the power line variations can be recognized by the timing pattern on the event system. Thus we have developed a new timing pattern aware EPICS controller which is called Pattern Aware Unit (PAU) for the beam line actuators (Fig. 1).

The PAU should be located in the actuator, and receives control values from the beam-based fast feedback or higher level feedback loop. The feedback system has multiple independent feedback loops which correspond to the each timing patterns. The PAU selects a set value from the multiple feedback loops according to the timing pattern, then implement the set value to the actuator.

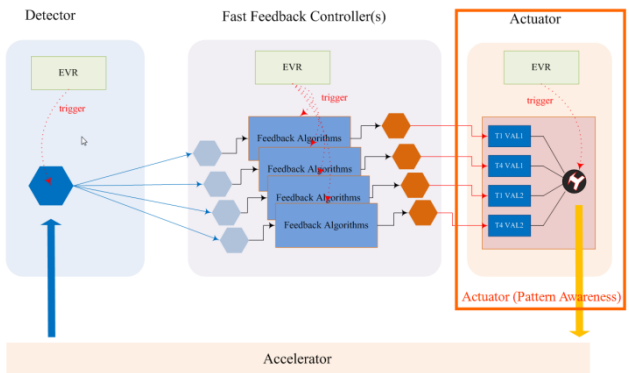


Figure 1: Concept of Pattern Awareness Unit.

REQUIREMENTS

The PAU is a generic software solution for the pattern aware operation. It is required to fit with the SLAC timing system and also required to work with the magnet control system, RF and any other actuator system which requires pattern aware operation.

The PAU is woken up by the fiducial interrupt from the timing system, and performs the pattern matching. The pattern is provided by the event system at every fiducial, and includes the following information: beam code, time slot, and 5 x 32 bits event masks [3].

The PAU performs the pattern matching two times: advanced pattern matching to set the actuator for the next beam pulse, and current pattern matching to get the control data from higher level feedback or to get the beam measurement data from the instruments. For the pattern matching, the PAU utilizes the event pipeline in the event system. This pipeline provides the pattern information two fiducial advanced. That is how the PAU takes action prior to the next beam pulse.

The PAU requires an accurate adjustable time delay due to the various delay times for the data gathering from the high level feedback and the beam measurement

*Work supported by the U.S. DOE Contract DE-AC02-76SF00515
#khkim@SLAC.Stanford.EDU

instrument. Thus, the PAU needs to handle the high-resolution timer on the CPU board. The PAU needs to support various actuators which have very different method to be handled. Thus, the PAU requires an extendability way to handle the different method for different actuators.

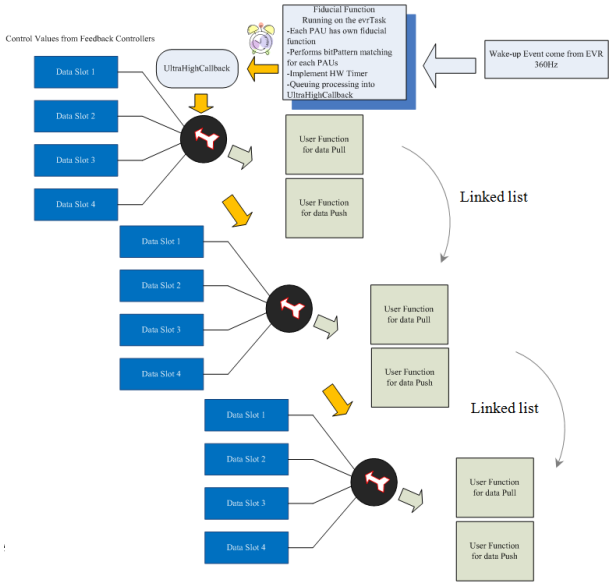


Figure 2: PAU processing for the Multiple Muxes.

IMPLEMENTATIONS

The event system provides a fiducial thread which is woken up by the 360 Hz fiducial interrupts. The PAU utilizes the fiducial thread for the advance/current pattern matching. Actually, the fiducial thread has a hook to register a user function, the PAU registers its hook function as the user function which has the pattern matching codes for the fiducial hook during the initialization. The pattern matching code performs the advance and current pattern matching with the event information from two fiducials ahead, and from the current fiducial, both of which areprovided by the event pipline in the event system.

If the PAU gets the matched pattern, the PAU sets up the high resolution timer on the CPU board which can be adjusted with sub nano-second resolution. The PAU has a user variable which is used for the adjustable time delay. User can configure the PAU delay with this variable. Actually, the delay value is decided by trial experiments to fit into the correct timing for the data gathering.

There is a high priority callback thread, the PAU's own thread called the UltraHighPriority Callback, which is woken up by the high resolution timer. This callback thread has a priority just lower than the fiducial thread and higher than any other EPICS thread. Thus, it can not be blocked by other tasks except the fiducial.

A PAU governs multiple multiplexers, or muxes, each mux corresponds to a physical quantity and a control variable: for example, RF phase, RF amplitude, or strength of corrector magnet. The multiple muxes in the same PAU share the same pattern matching. Thus, the PAU does the pattern matching once instead of the multiple muxes (Fig. 2).

This idea can reduce the pattern matching effort for individual control variable, and improves the real-time performance. This idea also saves the high-resolution timer which is a limited resource: there is finite number of high-resolution timer on the CPU board, in our case it has four. Each PAU occupies one high-resolution timer. Thus, it only allows maximum four PAUs on a CPU board with the high-resolution timer but, the mux idea allows it to handle a larger number of control variables. Actually, the muxes in a PAU has been implemented with the epics linked list (ellList), thus, there is no limit to create multiple instances.

The UltraHighPriority Callback visits each mux in the linked list one by one, and processes the user functions in the muxes. There are two kinds of user function: data pull

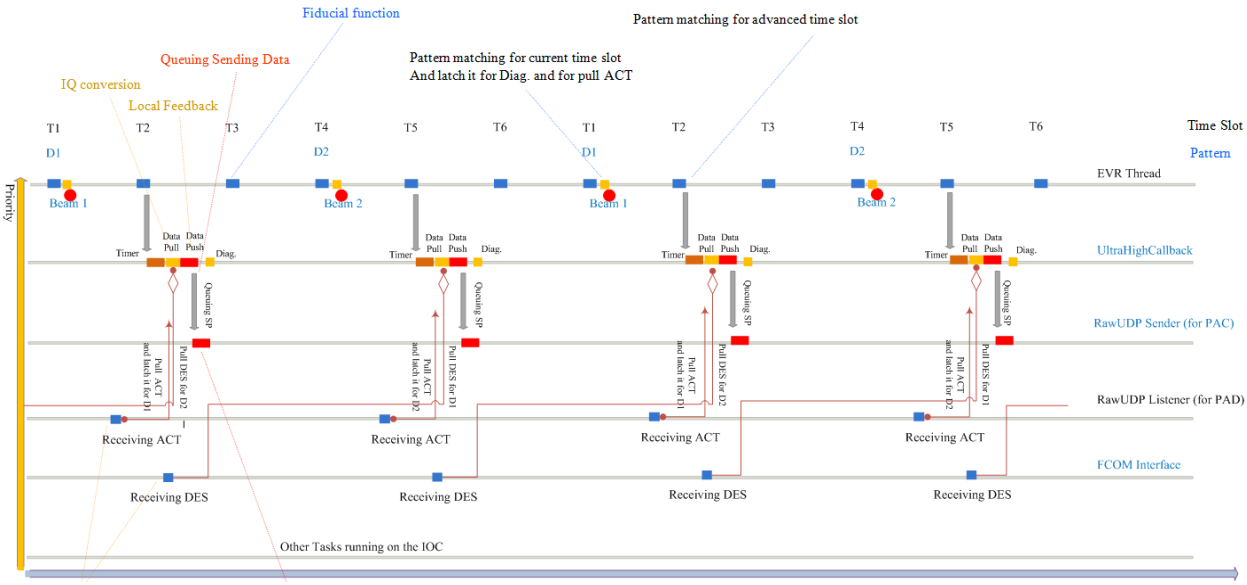


Figure 3: Time line for the 120 Hz Interlaced mode.

function for the current pattern matching, and data push function for the advanced pattern matching. System engineer who is using the PAU software for their system can register those functions for each of muxes.

The data pull function should perform data gathering from the higher level feedback or from the measurement instrument. The incoming data should be moved to the correct data slot which is decided by the current patter matching. The data pull function should perform set data to the actuator, this function brings the data from a correct data slot which is decided by the advanced pattern matching and implements the data into the actuator.

Some systems require more complexity for the data push function. For example, the RF system. This system has local regulation loop for phase and amplitude. The regulation loop requires additional calculations compare to others and also requires both physical quantities: phase and amplitude. Thus, we could not implement both muxes independently for phase and amplitude [4, 5].

The PAU has a concept namely 'Association' to make a relationship between both of the muxes, and we can use an iocsh command to make the association after creating the PAU and the muxes and before the ioc initialization. The concept for the association is not only used for making relation between phase and amplitude, also can be used for the more complex configuration. The local control loop for the LCLS laser RF system, there are three different sources: 2856 MHz oscillator, 119 MHz, and phase cavity. The local feedbacks for each source are related to each other to avoid jumping to a different bucket when switching it to a different source. We also utilize the association for the coupling for these three sources. The PAU has the other callback function namely diag function which has been implemented as a part of PAU software package and is performed very last time of the UltraHighPriority Callback.

After finishing every muxes' user function, the PAU processes a self-diagnostic function. This function does some house keeping work for PAU itself and measures the PAU performance: which data slot has been chosen by the pattern matching, how much time spent by the pattern matching, user functions including the self-diagnostic function itself, and what is the ISR delay for the high-resolution timer. The diag function updates the performance data every time when the PAU is activated by the fiducial, and this information is monitored by EPICS PVs.

Even if the beam based fast feedback supports the pattern aware operation, we still need to consider the non-pattern aware software. There is a longitudinal feedback system which has been written with MATLAB, and controls L2 and L3 section in LCLS, called 6x6 feedback. This 6x6 feedback does not support the pattern aware operation and is still being used for the LCLS. The static offset mode turns this non-pattern aware feedback into a pseudo-pattern aware (Fig. 4). The mux has offset PVs for the each data slot. When the mux implement the set value on the actuator, add up the offset value into the master set value. Thus, the non-pattern aware feedback provides the master set value, and operator or other software can provide pre-defined offset values for each patterns. The pre-defined offset value compensates differences between the different time-slots and patterns. Each mux has a PV to select the fast feedback mode (pattern aware) and static offset mode (pseudo-pattern aware).

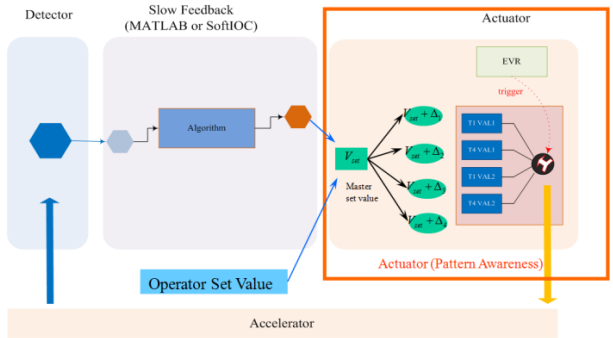


Figure 4: Static offset mode – backward compatibility for the non-pattern aware software.

APPLICATIONS

We have utilized the PAU software package for the RF system and magnets in the LCLS. These systems are controlled by the beam-based fast feedback system. The fast feedback system communicates with the actuator system through FCOM which is a dedicated communication protocol based on UDP/IP-Multicasting. The user pull function cares the FCOM communication with the fast feedback system, and the user push function works for the implement of the actuator set value.

Table 1: PAUs for RF System

PAU	MUXes and Associations	Remarks
PAU0 for Thales Laser System	1 station, 2 associations 3 muxes	2856 MHz PDES 119 MHz PDES Delegate PDES
PAU1for Coherent Laser System	1 station 2 associations 3 muxes	2856 MHz PDES 119 MHz PDES Delegate PDES
PAU2 RF Feedback for IN20	6 stations 6 associations 12 muxes	PDES/ADES for gun, L0A, L0B, TCAV0, L1S, L1X
PAU3 Feedback for LI24	7 stations 9 associations 18 muxes 2 virtual layers (L2/L3 abstraction)	PDES/ADES for L2Ref, TCAV3, KLY24_1, KLY24_2, KLY24_3, S29, S30, L2Abstr, L3Abstr

Especially, the user push function for the RF system is more complicated. It processes the local regulation loop for phase and amplitude, and network communication to the phase&amplitude controller (PAC). RF system has 4

PAUs and each PAU has various number of muxes. We need to use the association concept for the RF system, to make relation among different sources in the laser system, and to make relation between phase and amplitude.

We are using MATLAB based non-pattern aware longitudinal feedback for the L2 and L3 instead of the beam-based fast feedback. The PAU software provides pseudo-pattern aware for this feedback with the static-offset mode. We have implemented an abstraction layer between delegate muxes and individual klystron muxes. The abstraction layer controls the individual muxes to make a collective behaviour on L2 and L3 which reflects the delegate muxes. The non-pattern aware feedback is dealing with the delegate muxes. The abstraction layer has been implemented as the user functions for the muxes.

Table 2: PAUs for Magnet System

PAU	MUXes and Associations	Remarks
PAU0 Corrector in LTU0 area	1 corrector 1 mux	xcor_548
PAU1 Correctors in LTU1 area	3 correctors 3 muxes	xcor_488 xcor_493 xcor_593

PERFORMANCE MEASUREMENTS AND REAL-TIME DEADLINE

The PAU has self-diagnostics which provides accurate measurements for processing time and delay for the every important step. We have utilized this self-diagnostics for the real-time deadline analysis. Table 3 shows the processing time, the worst case is RF system and it takes ~ 105 µsec. According to the Figure 5, PAU wakes up 2 fiducials prior from the next beam. But, PAU need to wait ~ 200 µsec for the communication delay from the beam-based fast feedback or higher level feedback. Some of the actuators require a settling time, worst case is magnet. It requires almost 6 msec. We can consider fiducial interval 2,778 µsec and beam delay from fiducial ~ 1 msec, then the PAU has 251µsec margin from the real-time deadline.

Table 3: Real-Time Performance Measurement

PAU Processing	Processing Time or Delay
Pattern Matching in Fiducial	2 µsec
ISR delay	3 µsec
User Pull Function	25 µsec
User Push Function	60 µsec
Self-diagnostics and house keeping	15 µsec

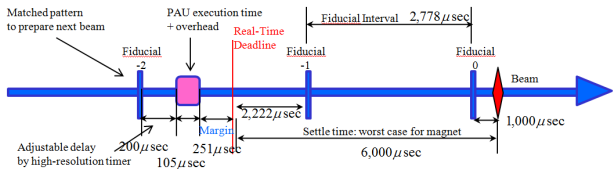


Figure 5: Performance Measurement and Real-Time Deadline.

SUMMARY

The PAU software has been designed as a generic software module for the LCLS beam line actuator. It was implemented to get adaptability to fit into various systems which have particular and unique requirements. We applied PAU software to the RF system, and magnet system, and it contributed for the stable 120 Hz beam-rate operation on LCLS. During this work, we accomplished the followings.

- Integrate with the beam-based fast feedback system for the RF system, and Magnet System
- Achieve the pattern aware operation
- Support non-pattern aware longitudinal feedback for L2 and L3, and make it to pseudo-pattern aware operation
- Implement the various user pull/push functions for the particular/unique requirement: RF local regulation loop, Amplitude/Phase to I and Q conversion, Abstraction Layer for L2 and L3.

REFERENCES

[1] D. Fairley et. al., "Beam Based Feedback For the Linac Coherent Light Source," ICALEPCS 2011 in Grenoble, October 2011.

[2] P. Krejcik and et. al., "LCLS Timing System Requirements," LCLS Physics Requirements Document #1.1-305, 2005.

[3] S. Allison, "Timing and Event System for the LCLS Electron Accelerator," EPICS Collaboration Meeting in Vancouver, May 2009.

[4] A. Akre and et. al., "RF Control Requirements," LCLS Engineering Specific Document #1.1-301, 2005.

[5] D. Kotturi, "LCLS LLRF Distributed Control System," EPICS Collaboration Meeting in Vancouver, May 2009.

YAMS: A STEPPER MOTOR CONTROLLER FOR THE FERMI@Elettra FREE ELECTRON LASER*

A. Abrami, M. De Marco, M. Lonza, D. Vittor, Sincrotrone Trieste, Trieste, Italy.

Abstract

New projects, like FERMI@Elettra, demand for standardization of the systems in order to cut development and maintenance costs. The various motion control applications foreseen in this project required a specific controller able to flexibly adapt to any need while maintaining a common interface to the control system to minimize software development efforts. These reasons led us to design and build "Yet Another Motor Subrack", (YAMS), a 3U chassis containing a commercial stepper motor controller, up to eight motor drivers and all the necessary auxiliary systems. The motors can be controlled locally by means of an operator panel or remotely through an Ethernet interface and a dedicated Tango device server. The paper describes the details of the project and the deployment issues.

INTRODUCTION

The starting point for this project was the history of the ELETTRA storage ring and particularly of its beamlines and experimental chambers where, although the motion problem was central for experimental people, a number of different solutions were used, ranging from “commercial off the shelf” to “entirely home made”. This was the stimulus and the knowledge background from which this stepper controller has been designed.

Project Design Criteria

One of the design goals was to realize a modular and “reasonably” flexible device suitable for the major number of use cases in FERMI@Elettra motion problems. “Reasonably” in the sense that we didn't want to replicate any of the market available controller that claimed to fit with a lot of (all) plants. The reference motor type was fixed as two-phase one.

The initial idea was to assemble the core controller board, a suitable motor power supply and the motor driver boards into a 3U size subrack.

A motor driver board was designed to fit into a DIN41494 Eurocard standard, so a “per motor” modularity could be achieved. It was foreseen to host a small daughter board dedicated to encoder signal conditioning, thus realizing the necessary “encoder flexibility”.

The ELETTRA motion control history led us to use the Galil DMC-21x3 as the core of this project [1]. This family of motor controllers may drive from one to eight axis. In order to assure flexibility, the internal connections between the DMC and the drivers have been split in two groups of four axis each.

An obvious effort was done to standardize the connections to motors and encoders: one connector for the “axis”, i.e. motor phases, limit/home switches (with supply for optical ones), brake control and “emergency stop” signal; and one connector carrying signals both for “digital” and “analog” encoders.

Another design criteria was to be able to re-use this stepper motor controller also in the ELETTRA beamlines, which will anyway require the design of new 3 and 5 phase driver boards.

Software application programs availability, especially in the Tango control framework, was another key point in the enumeration of the criteria used in the design phase.

Figure 1: YAMS chassis (front view).

HARDWARE DESCRIPTION

A 3U subrack contains the power supply for the motor driver circuitry, the DMC controller, the power supply for the DMC and for the other electronics (encoders, etc.), a simple local control panel and the Ethernet connection. In addition, this subrack may be equipped with up to 8 motor driver boards.

Figure 2: YAMS chassis (rear view).

In Fig. 2 a YAMS subrack equipped with six motor driver boards is shown; blind panels are used to close the unused slots (here not shown).

*Work partly supported by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3

YAMS Subrack

Table 1 summarizes the main characteristics of this controller:

Table 1: YAMS Subrack Characteristics

Function	Value
Mains power supply	220V
Motor power supply	24V, 350W max
Electronics power supply	5V, 50W - ±12V, 30W
Core motion controller	DMC family by Galil Inc.
Communication Interface	Ethernet 10BASE-T
Local Interface & Control	Motor, speed (1 out of 2), direction-movement selectors Stop-all-motion pushbutton Power supply leds.
Motor driver boards	up to 8

YAMS Motor Driver Boards

These boards are designed in the DIN41494 Eurocard standard, each carrying the electronics necessary to cope with different motor type and power. From the point of view of the core controller the connecting bus is the same, being its main feature the pulse/direction interface. The power section of the board is realized using the IMxxxH hybrid family produced by IMS (now "Schneider Electric Motion USA") [2].

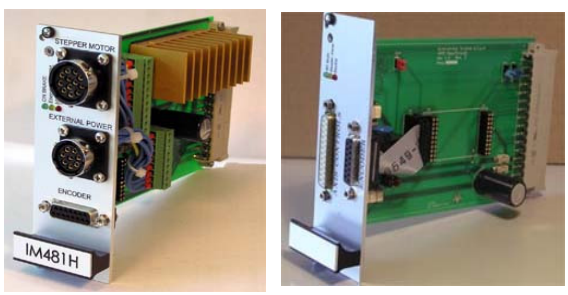


Figure 3: (A, left) Low Power Driver Board IM481H, (B, right) Passthrough Driver Board.

Table 2 shows the main common characteristics of these boards:

Table 2: YAMS Motor Driver Characteristics

Common feature	
Motor power supply	24V
Electronics power supply	5V, ±12V
Motor type	2 phase
Motor run/hold current configuration	via dip switch

As Fig. 3A shows, the board features three connectors. The first (upper in the figure) is dedicated to the axis connections: motor phase, limit/home switches (with power supply for the optical ones), brake and emergency stop signal. The central connector is used for connecting the hybrid power, i.e. the power section of the board, to the internal power supply, or to an external one. In this way multiple high power motors may run at the same time. In the third connector, the sud-D one, there are all the encoder signals, both for digital and analog encoders.

Table 3 summarizes the motor current ranges of the three adopted IMS boards:

Table 3: YAMS Motor Current Ranges

Card Name	IM481H	IM483H	IM805H
IMS hybrid	IM481H	IM483H	IM805H
Motor current (peak)	0.2– 2.1 A	0.5 – 4.2 A	1 – 7.1 A

Besides the driver board that actually may power a stepper motor, a “versatile board” was designed called “Passthrough”, Fig. 3B. This board is equipped with two connectors. One of them carries the motor, switches and brake signals: this is not directly connectable to the phases of a motor, a power stage with the pulse/direction interface must be interposed (for instance motors with power electronic integrated). The other connector is exactly the same sub-D encoder connector of the other boards, and, in fact, this board has the same “encoder conditioning philosophy” and may host the same range of encoder daughter boards.

YAMS Encoder Daughter Boards

The purpose of the encoder boards is to adapt and/or condition the signals coming from the encoder to the inputs accepted by the DMC controller. Up to now we have developed three types of encoder boards, as shown in Fig. 4:

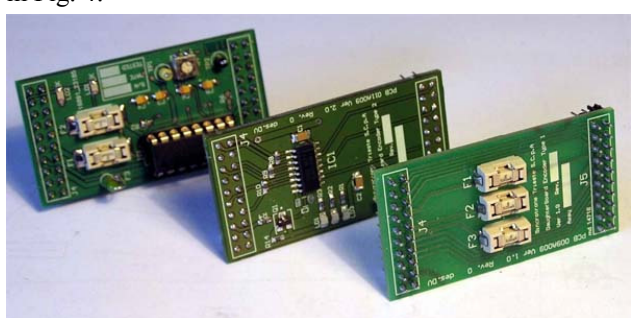


Figure 4: Encoders Boards.

- Type 1 encoder board: it just connects the external encoder connector to the internal signal buses. No conditioning is performed.
- Type 2 encoder board: it converts RS422 level balanced signals from digital sin/cos encoders to unbalanced signals.
- Type 3 encoder board: it accepts a 4-20mA analog signal and converts it to an internal voltage signal.

SOFTWARE DESCRIPTION

The DMC family controllers have their own communication and programming protocol and language, besides a good basic instruction set for motion control. So, if necessary, it is possible to add specific routines into the controller memory to improve the controller functionalities. These routines may be referred as "firmware".

A similar solution for motion control was also adopted by the SOLEIL synchrotron light source, participating with other partners, including Sincrotrone Trieste, to the Tango collaboration. They developed the software architecture and the first releases of the "galilbox-srv" and "galilaxis-srv" Tango device servers (Fig. 5). As a result, the YAMS project has inherited a considerable amount of knowledge and ready to re-use code, thus reducing the overall development time.

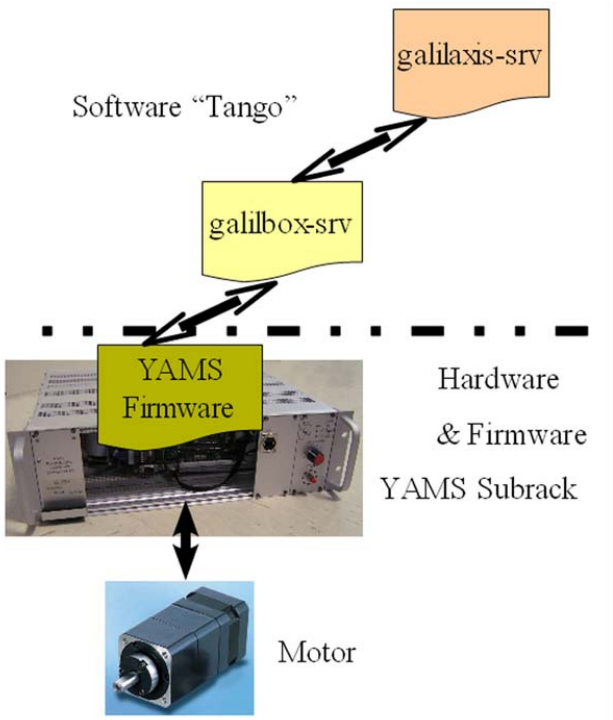


Figure 5: Hardware/Software Architecture.

Differences between SOLEIL and FERMI@Elettra in the modality the stepper controllers are employed, led us to modify both firmware and software from their initial releases. For instance, the FERMI@Elettra version is capable to read a potentiometric encoder, i.e. an analog sensor, and close on it the motion control loop.

FERMI INSTALLATIONS

In the whole facility, more than 400 axes will be eventually controlled by YAMS, using different phase currents, with or without brakes and with different encoder types, RS422 digital, potentiomentric and LVDT. The range of applications span from heavy duty ones (like the "bunch compressor" in Fig. 6), to high precision ones such as the photon spectrometer installed in the photon beam transport area (Fig. 7).

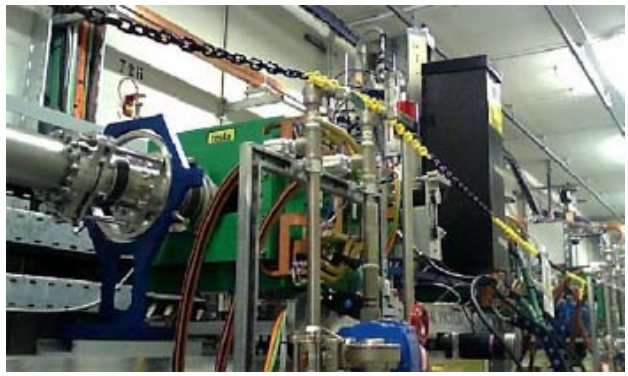


Figure 6: Bunch Compressor, an heavy duty installation.

Figure 7: Photon spectrometer, an high precision installation.

DEVELOPMENT

A new driver board is under development, it will adapt the YAMS bus interface to the old "Berger-Lahr" D450 (5 phase) and D920 (3 phase) stepping motor cards, thus confirming and realizing the "retro-fit" issue for the old installations in ELETTRA.

A piezo motor driver board and a 1Vpp analog encoder daughter board will soon be designed for beamlines and experimental chambers.

CONCLUSIONS

In this project, a significant reduction of the development time has been achieved, meanwhile maintaining costs at a reasonable low level. The "in depth" knowledge of all hardware and software components of this motion system allows to suggest the best solution for each motion request that arose during the design and installation. In the installations carried out up to now, YAMS has proved the good features of modularity and flexibility of this motion approach.

REFERENCES

[1] DMC-21x2/21x3 User Manual Galil Motion Control, Inc. http://www.galilmc.com.

[2] "PCB mounted microstepping drivers", IM481/483/805H IMS http://imshome.com.

COMPARATIVE ANALYSIS OF EPICS IOC AND MARTe FOR THE DEVELOPMENT OF A HARD REAL-TIME CONTROL APPLICATION*

A. Barbalace†, A. Luchetta, G. Manduchi, C. Taliercio, Consorzio RFX, Padova, Italy
B.B. Carvalho, D.F. Valcárcel, IPFN, Lisboa, Portugal

Abstract

EPICS is used worldwide to build distributed control systems for scientific experiments. The EPICS software suite is based around the Channel Access (CA) network protocol that allows the communication of different EPICS clients and servers in a distributed architecture. Servers are called Input/Output Controllers (IOCs) and perform real-world I/O or local control tasks. EPICS IOCs were originally designed for VxWorks to meet the demanding real-time requirements of control algorithms and have lately been ported to different operating systems.

The MARTe framework has recently been adopted to develop an increasing number of hard real-time systems in different fusion experiments. MARTe is a software library that allows the rapid and modular development of stand-alone hard real-time control applications on different operating systems. MARTe has been created to be portable and during the last years it has evolved to follow the multi-core evolution.

In this paper we review several implementation differences between EPICS IOC and MARTe. We dissect their internal data structures and synchronization mechanisms to understand what happens behind the scenes. Differences in the component based approach and in the concurrent model of computation in EPICS IOC and MARTe are explained. Such differences lead to distinct time models in the computational blocks and distinct real-time capabilities of the two frameworks that a developer must be aware of.

INTRODUCTION

Developing a real-time control application is a complex task if it has to be designed from the ground up. Everything must be checked and carefully reviewed to guarantee bug-free code and bounded execution times. Likely there exist many different programming environments to code a real-time control system and many are suited for scientific experiments. In this work the EPICS [1] and the MARTe/BaseLib [2] (in the following referred as MARTe) software frameworks are analyzed.

The EPICS software suite is built around the Channel Access (CA) network protocol that allows the communication of different EPICS clients and servers in a distributed architecture. Servers are called Input/Output Controllers (IOCs) and perform real-world I/O or local control tasks.

MARTe is a software library that allows the rapid and modular development of stand-alone hard real-time control applications on different Operating Systems (OSs).

As far as we know there are no published comparative analyses of software frameworks for the development of control applications, especially involving EPICS and MARTe. This work is intended to complement an earlier paper [3] where we presented a performance comparison of the two software frameworks.

We analyze EPICS version 3.14.11 and MARTe CVS version June 2011. All facts presented here are OS-independent, whereas paper [3] refers to Linux PREEMPT_RT. In this paper we address several implementation differences and similarities of the two software frameworks which are both written in the C/C++ language.

ARCHITECTURAL OVERVIEW

Component Based Approach

Both EPICS IOC and MARTe present a modular architecture where components can be connected to carry out the required control algorithm. Components in EPICS IOC are implemented by *records*, while in MARTe they are implemented by *Generic Application Modules* (GAMs). Records are logically grouped in *databases*, GAMs belong to a *Real Time Thread*.

In EPICS new record types (or *Record Supports*) can be created by providing a set of routines and data structures adhering to a given C prototype; the newly created record will carry out the required computation. For every record type it is possible to define a new *Device Support* that enables the record to interact with an I/O device.

New components in MARTe are created by subclassing the GAM class or any other descendant of the GAM class. By subclassing, all the functionalities required to properly interact with the MARTe environment are inherited by the new component. In MARTe only two special subclasses of the GAM can handle data communication with a device driver: `InputGAM` and `OutputGAM`. Such classes can be associated to an I/O device via a *Generic ACQuisition Module* (GACQM), similar to what is called Device Support in an EPICS IOC.

Configuration Capability

The current configuration of both frameworks, i.e. the sets of component instances and the way they are intercon-

* This work was set up with financial support by Fusion for Energy.
† telephone: +39 049 8295039, e-mail: antonio.barbalace@igi.cnr.it

nected, is defined in one or more text files which get parsed to produce the target application.

In an EPICS IOC the records that populate a database are declared in a configuration file. In MARTe a configuration file contains not only the functional blocks (GAMs) of the control algorithm but also instance descriptions of generic C++ classes. Such instance descriptions can be specified to extend the functionality of the target MARTe application. As an example, to monitor the data exchanged by the GAMs in a control loop an instance description of a Web Server class can be included in the configuration file.

In both EPICS and MARTe a control application is described by a configuration file. The way in which such files are used to create the application differs between the two frameworks.

Creating a New Application In EPICS a configuration file is used to generate IOC source code before compilation. Instead, in MARTe, a configuration file is loaded by a running MARTe system.

Both frameworks come with a set of base components but the developer may write its own components (records or GAMs). When a new record is ready to be tested in EPICS the developer has to create a new project, insert the record with all ancillary files in the project, create the configuration files, compile and run the project. In MARTe the development process is straightforward: after compiling the new component the developer has only to write a configuration file (actually by modifying a template).

Updating a Running Application The value and compound data (database links included) of a record can be tuned at runtime being each field implemented by a Process Variable (PV). To load or remove records not previously loaded in EPICS a user has to reconfigure and recompile all the application from scratch and then run it again.

In MARTe in order to change a GAM parameter it is necessary to stop the execution of every MARTe activity and reload the framework. It is however not necessary to recompile the application. A new component which allows the runtime modification of parameters and signal values called *Configuration Library* (CL) [4] was recently introduced.

Internal Data Structures

In EPICS IOCs every record field is a PV, and in order to manage thousand of PVs, EPICS makes use of hash tables for PV lookup.

MARTe is mostly based on linked lists and sophisticated data structures have not been introduced. Data querying at runtime is still lacking.

Both frameworks create during initialization all data structures that are required at runtime. It is the responsability of the developer to implement efficient (time-bounded) code in the `process` routine (EPICS IOC) or in the `execute` method (MARTe) of a computational block. In a hard real-time application unbounded delays due to badly written algorithms or network accesses could deteriorate the control.

CONCURRENT MODELS OF COMPUTATION

Concurrent models of computation (CMoC) are of great interest when dealing with embedded systems, especially feedback controllers. A model of concurrence specifies how interaction, communication and control flow will be handled between components in a software application. In the following the models of computation of EPICS and MARTe are described.

EPICS IOC Record Processing

A single record in an EPICS IOC can be accessed concurrently for reading, writing or processing. Reading and writing can trigger processing, before fetching a value (demand-driven) and after updating a value (data-driven).

The processing of a chain of records takes place within a *scan* (an operating system's thread). A scan can be locally triggered periodically (*periodic scan*) or by software and hardware events (*event scan*, *IO event scan*); remotely by `caGet` or `caPut` operations. After the processing has been triggered on the first record of a chain those records that have to be executed next are determined by the associated *links* in the database. A link can carry data and processing (INLINK, OUTLINK) or simply processing (FWDLINK). A chain of records is executed until there exist processing links or records which can be passively processed by other scans (*passive scan* in EPICS terminology). Very complex processing chains can be designed in EPICS mixing demand (INLINK, FWDLINK) and data (OUTLINK) -driven processing.

Because a single record can be concurrently accessed by different scans, per-record locks exist. Synchronization delays due to concurrency can deteriorate the real-time response of a control algorithm. Typically such a situation arises when a record is concurrently updated locally, belonging to a processing scan chain, and remotely by a CA operation (Fig. 1). The local operation issued by the EPICS IOC has lower priority then the remote operation.

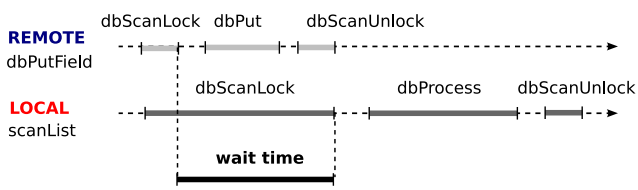


Figure 1: In EPICS a locally triggered scan may need to wait for the end of the processing of a remotely invocated record.

MARTe GAM Processing

MARTe has been designed as a software framework for hard real-time, low-latency applications. It eliminates concurrency because it is a source of delays and uncertainties in code execution. MARTe is a lock-free execution environment where each GAM executes serially in a thread (`RealTimeThread`) to which it is uniquely tied. MARTe has been designed to support a multiprocessor environment and the developer is encouraged to map a RealTimeThread per CPU.

GAMs communicate only within thread bounds where communication is made safe by the inherently serial execution of the thread. Inter-thread communication exists, but is managed by special GACQMs. GAMs communicate by means of a thread-level data buffer called *Dynamic Data Buffer* (DDB); every GAM knows how to access their private data area. This model implies a strictly data-driven processing.

As mentioned in the previous section, parameter and signal values can be modified via CL. Remotely changing a value in a running MARTe application is possible and is completely asynchronous with the execution of the RealTimeThread (highest priority task).

Processing Details

Stack Frame MARTe is less memory resource-starved than EPICS. A long EPICS' chain of passive records with many links can produce an out-of-stack exception, while MARTe will never exhibits this problem because the number of execution frames on the stack is $O(1)$ while in EPICS it is $O(n)$ where n is the number of passive chained records.

Processing Links Links in MARTe define only the flow of data and therefore they establish the data dependency between computational blocks. In an EPICS IOC links define the flow of data and processing.

Data Conversion at Interfaces Data diversity between blocks can be handled by the framework to let blocks with different data interfaces communicate. Data conversion can be managed at the node level via polymorphic get/put routines (Fig. 2) or by means of an external (global) adapter (Fig. 3). EPICS supports data diversity by means of a rich set of protocol adapters; MARTe does not support data diversity: two communicating GAMs must adhere to the same data interface.

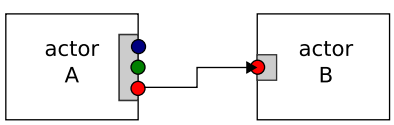


Figure 2: Actor A shown a polymorphic interface: the same **get** method will be overloaded to return different data types in function of the data type required by actor B.

Fanout In electronic circuits the output from one Integrated Circuit (IC) can be connected to several ICs. The fanout indicates how many ICs can be connected to a single output. The same concept also applies to records and GAMs. In MARTe one output signal from a GAM can be connected to as many GAMs as needed.In EPICS it is possible to have many INLINKs to the same record's field but it is not possible to have an OUTLINK or a FWDLINK connected to more than one record's field; to fulfill this need a `fanoutRecord` is supplied with EPICS.

Data Tokens In both software the processing is due to a new data token and there is no buffering between nodes and the number of tokens to be consumed in any processing of a node is always one.

CONCURRENT MODELS OF TIME

The natural way to code a control algorithm is via discretization of a continuous time model. The discretized algorithm is not aware of time, moreover it is usual practice to maintain a discrete function $f(n)$ instead of $f(Tn)$ where T, the sampling period, is explicit in the formula. The algorithm runs after the samples are transferred from the data acquisition subsystem, once per sampling period. The period T is usually fixed a priori during the design phase.

Coding the Algorithm in MARTe

In MARTe each GAM is triggered once per period and the time elapsed between calls to GAMs is equivalent to period T. Periodic timing is provided by a GACQM interfaced with a `TimeInput` GAM. Algorithmic GAMs are aware of the absolute time and of the execution period T.

Coding the Algorithm in EPICS

EPICS' records can be processed concurrently and lock mechanisms have been developed to synchronize their processing. In EPICS no time model is assumed, a record can be processed at any time. Careful design of an EPICS IOC database is required to achieve periodic processing of a record. To develop a record, a user have to take into account the possibility that a discrete algorithm will be executed in a non-uniform sampling environment.

The smart way chosen in EPICS to develop a PID controller (`pid`, `cpid` and `epid` records) is to accept that the

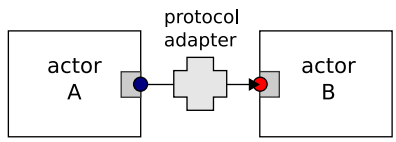


Figure 3: Both actors have a rigid interface: a protocol adaptor is necessary. A's **get** method returns a unique data type and B's **put** method fetches a precise data type.

PID will be called at non-uniform time intervals, and to define a minimum amount of time between record processings. This is defined in the Minimum Delta Time (MDT) field of the PID record. If the amount of time between the last time the record was processed and the current time is less than MDT, then the record is not processed.

REAL-TIME SUPPORT

Both EPICS IOC and MARTe have been used on VxWorks installations but this is not enough to state that they are hard real-time software frameworks. In [3] we have compared the same test control loop implemented as an EPICS IOC and as a MARTe application on a Linux PREEMPT_RT system. EPICS IOC exhibits a wider jitter compared to the MARTe counterpart. In this section we figure out from where those sources of jitter come.

Event Triggering

In MARTe the way in which a device driver (GACQM) waits for a hardware event is selectable (polling or interrupt) and defined in the configuration file. In EPICS the developer can not choose how to wait for an event, it depends on the device driver. Polling may be advantageous because it can eliminate the OS's process switching (which could introduce further delays due to scheduling other tasks).

Event Dispatching

In EPICS every event is queued for execution and never lost. There are different queues of execution and the execution policy is FIFO. One queue handles CA requests; three queues (low, medium and high priority) are shared between events and IO events and there is one queue per periodic scan (Fig. 4). This mechanism decouples an event from the associated servicing code but can generate unbounded latencies. Different scans that originate from different interrupt sources but with the same priority are enqueued on the same queue. Such an approach can introduce considerable jitter in the generated waveforms and task deadline misses. MARTe does not queue any event: events can be lost but there is no jitter due to events enqueuing.

Exploiting Multiprocessors

MARTe allows the developer to assign threads and IRQs to specific processors. This facility protects the execution and promotes a deterministic response of the thread. The affinity mask of threads and IRQs on the processors is a parameter in the MARTe configuration file.

The EPICS Operating System Interface (OSI) layer does not support multiprocessor environments, i.e. `epicsThreadCreate` does not accept any CPU affinity argument. A static set of queues is defined for each instance of EPICS IOC. This approach does not scale to multiprocessors.

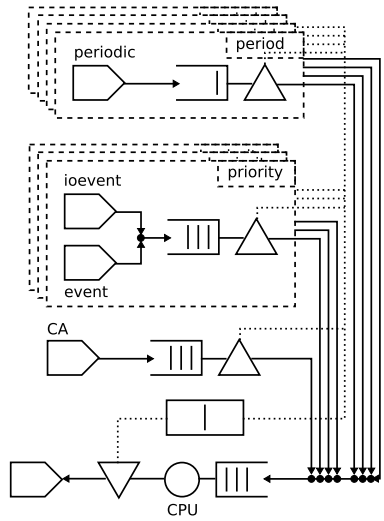


Figure 4: **EQN model** of the EPICS events queuing system running on a single processor/core machine.

FINAL CONSIDERATIONS

Since EPICS has been on the scene since 1989, the number of tools available to design and configure a system has grown. The same does not hold for the real-time support and especially for the exploitation of emerging processor technologies (multiprocessor architectures). Coding a discretized control algorithm can be tricky.

MARTe, on the other hand, has so far only a limited set of design and configuration tools. Some work is ongoing to exploit the Ptolemy [5] project as a tool for configuring and simulating MARTe applications. MARTe appears to be crafted for real-time, low latency applications. To cope with such demanding requirements it exploits emerging multi-core architectures.

REFERENCES

[1] A. Johnson. (2007) Epics about [Online]. Available: http://www.aps.anl.gov/epics/about.php

[2] A.C. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, L. Zabeo, A. Barbalace, H. Fernandes, D.F. Valcárcel, and A.J.N. Batista, "MARTe: A Multiplatform Real-Time Framework", *IEEE Transactions on Nuclear Science*, vol. 57 no. 2, pp. 479 – 486, Apr. 2010.

[3] A. Barbalace, G. Manduchi, A. Neto, G.D. Tommasi, F. Sartori, and D.F. Valcárcel, "Performance comparison of EPICS IOC and MARTe in a hard real-time control application", *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pp. 1 – 5, May. 2010.

[4] D.F. Valcárcel, A. Barbalace, A. Neto, A.S. Duarte, D. Alves, B.B. Carvalho, P.J. Carvalho, J. Sousa, H. Fernandes, B. Gonalves, F. Sartori, and G. Manduchi, "EPICS as a MARTe configuration environment", *IEEE Transaction on Nuclear Science*, vol. 58 no. 4, pp. 1472 – 1476, Aug. 2011.

[5] U. B. E. Department. (2011) Ptolemy project home page [Online]. Available: http://ptolemy.eecs.berkeley.edu/

DEBROS: DESIGN AND USE OF A LINUX-LIKE RTOS ON AN INEXPENSIVE 8-BIT SINGLE BOARD COMPUTER

Mark A. Davis, NSCL, East Lansing, MI 48824, U.S.A.

Abstract

As the power, complexity, and capabilities of embedded processors continue to grow, it is easy to forget just how much can be done with inexpensive Single Board Computers (SBCs) based on 8-bit processors. When the proprietary, non-standard tools from the vendor for one such embedded computer became a major roadblock, I embarked on a project to expand my own knowledge and provide a more flexible, standards based alternative. Inspired by the early work done on operating systems such as UNIX™, Linux, and Minix, I wrote DEBROS (the Davis Embedded Baby Real-time Operating System), which is a fully pre-emptive, priority-based OS with soft real-time capabilities that provides a subset of standard Linux/UNIX compatible system calls such as stdio, BSD sockets, pipes, semaphores, etc. The end result was a much more flexible, standards-based development environment which allowed me to simplify my programming model, expand diagnostic capabilities, and reduce the time spent monitoring and applying updates to the hundreds of devices in the lab currently using such inexpensive hardware.

INTRODUCTION

Since I first joined the Control Systems group at the NSCL, my primary focus has been on designing and developing software for embedded controllers. My first project was to upgrade several existing controllers from an in-house SBC design based on the Motorola 68701 micro-controller. These controllers used a low-speed RS-485 multi-drop network that resulted in frequent communication outages whenever someone disconnected one of the controllers or one of the connections failed. The goal of the upgrade project was to replace the RS-485 networks with a point-to-point Ethernet network.

To that end, an inexpensive SBC was chosen that incorporated the 8-bit Rabbit 2000 micro-controller and a 10base-T Ethernet interface. The SBC was the BL2105 from Z-World, Inc. (later purchased by Rabbit Semiconductor, and now part of Digi International Inc). This SBC includes 512K of RAM and 512K of flash memory and runs the Rabbit processor at 22.1MHz. At the time, this was a huge step up from the 68701 design which had only the built-in 128 byte RAM, 2K UVEPROM, and a top speed of 2MHz.

Another major improvement in this move was the change from programming entirely in assembly language to writing almost all the code in C.

With an Ethernet interface included on the SBC and a library of functions to support IP networking, the change in the communications interface was not a difficult task. Once that was accomplished, it was time to consider the many other enhancements made possible by the much more capable hardware.

EARLY ENHANCEMENTS

Compared to the Rabbit-based SBC, the earlier 68701 design had some major limitations:

- The tiny memory severelly limited code size
- Modifying UVEPROM requires physical access
- All code was written in assembly language

Given these limitations, there was no way to do regular firmware upgrades, much less add functionality beyond the bare requirements.

But the new hardware did not have these limitations. With vastly more RAM, software rewritable persistent storage (flash), and a high-speed communication interface, increased functionality and frequent updates were now possible. As the number of controllers and the complexity of their software grew, it quickly went from being possible to being necessary.

And so it began: More processing power at the controller level meant the controllers could do more of the work and could respond much faster to changes than the control system could. The size and complexity of the code on the controllers grew exponentially, which required additional diagnostic capabilities (adding yet more code) and an efficient way to update the firmware on the rapidly growing number of controllers.

It wasn't long before the inevitable happened: The new controller's larger memory space was used up, and there was no more room for enhancements.

A SQUARE PEG IN A ROUND HOLE

Several factors contributed to our problem of not being able to continue expanding the capabilities of the new controllers, including imitations in the vendor's development tools. But it also became apparent to me that a more basic issue needed dealing with: The programming model I was using was resulting in large, complex code that was difficult to maintain and was eating up memory much too quickly. Something needed to be done.

At the time I was using a cooperative multitasking model to keep my code organized in to separate "tasks". Scheduling consisted of the simple round-robin method of a series of function calls inside an endless loop, with each function representing one task.

In the simplest cases, each function will run from beginning to end each time it is called. This "run-to-

completion" model can be very efficient. It can also greatly simplify concurrency issues because no "task" is every pre-empted.

The problem with the run-to-completion model is that the average response time of the system for any single function grows larger and larger as the number of functions and/or the time spent in each function grows. When the average (or maximum) time for one pass through the main loop becomes too long, you are forced to break functions into smaller and smaller pieces to maintain acceptable response times.

At that point, the simplicity of the run-to-completion model falls apart. Breaking each function in to smaller and smaller parts and executing just one part each time the function is called stops the average time between calls from growing too large. But it also means it takes longer for each function to do the work it previously did during a single call.

In addition, breaking each function into pieces only works if each piece can pick up where the last one left off. It means every piece must be responsible for maintaining the internal state of the function between calls.

So every function becomes larger, slower, and more complex, even if you aren't adding any functionality to it. And each additional function or enhancement to an existing one potentially requires you to adjust the number, size, and/or content of the pieces in other functions.

And this doesn't even address the issue of different priorities for each task, and how to minimize the time between when an event occurs and when the function that needs to process it is called.

Eventually I realized that this was not a sustainable model. To run smoothly, the expanded capabilities of the controller required the behaviour of a pre-emptive, priority-based multitasking kernel, and I was trying to achieve that using a cooperative multitasking model. The result was that every task had to take on the responsibilities that are normally handled for them by a pre-emptive kernel. Once I realized this, it was painfully obvious why my code had become so large and complex. The question now was what could I do about it?

A PARADIGM SHIFT

At this point I knew a fundamental change was needed. The obvious alternative to the cooperative multitasking model was the pre-emptive one. If I had a pre-emptive kernel, there would be no need for every task to do the kernel's job. The mass of complex code I had to add to each task could be removed, which would free up a lot of memory and make them run faster. But of course, recognition of a problem doesn't automatically mean there is a practical solution.

When considering my options, my first thoughts were of MINIX and Linux. Even early versions of these operating systems where "complete" in the sense that, in additions to having pre-emptive, multitasking kernels, they also supported login shells and user commands. And I knew from personal experience that the original version of MINIX could run reasonably well on a 4.77 MHz IBM PC with only 256K memory. So I was confident that what I needed was possible.

Unfortunately, even the smallest Linux distributions I could find were far too large for my use. I briefly entertained the idea of trying to locate an electronic copy of the original source code for MINIX, but the design of MINIX varied in some very significant ways from what I needed (e.g. it used run-time loaded re-locatable binaries and a message-passing microkernel, both of which pose problems for real-time scheduling). [1]

Having failed to find a "complete" solution (a kernel, user interface, support for sockets), I decided the next best thing would be to use a pre-made kernel and write or port the rest of what I needed.

DEBROS IS BORN

I knew that, even if I found a suitable ready-made kernel that supplying the rest was going to require a lot of work. But in addition to the very real need for a more practical runtime platform, I was motivated from the start by the desire to learn about how pre-emptive multitasking actually works. When I was getting my degree in Computer Science the primary focus was on formal languages and writing compilers. There is no end to how useful the insights I gained from that knowledge have been over the years. But I am finding that the things I learned while doing this project have been just as fundamental to how I think about programming. In retrospect I am surprised that this topic was all but ignored in my college courses.

So I eagerly began my new programming and self-learning project by purchasing a pre-made kernel named TurboTask from Softools, Inc. along with a copy of their compiler.

TurboTask is a very small and efficient kernel that supported all the basic features I needed. The Softools compiler and linker produced smaller and faster code than the hardware vendor's tools, and included a copy of the hardware vendor's network stack (whose API I was already familiar with). All my early development was done using the TurboTask kernel, and the first version of DEBROS used on controllers put into production was one that incorporated it.

Unfortunately, as things progressed, it became apparent that the TurboTask kernel was not as good a fit as I had hoped it would be. It was indeed small, fast, and highly optimized. But it really was designed for someone programming at a much more hardware-oriented level. It just didn't mesh well with the needs of the UNIX-style OS I was developing. One limitation in particular - the requirement that the stacks for all tasks had to reside in the first 64K of memory – became a deal-breaker.

Eventually I had to replace the TurboTask kernel with my own task switcher, which was itself derived from an example provided by SHDesigns, the firm that sells the Download Manager/boot-loader we now use. The Softools compiler and tools, however, have continued to be critical to the success of DEBROS.

Regardless of the implementation details, by this time I had achieved my primary goal: The individual tasks no longer had to do the work of a kernel. I was able to strip out all the extra code that had nothing to do with each task's primary purpose. The resulting reduction in size and complexity was so significant it was if a thick fog had lifted and revealed a sunny sky.

But as important and fundamental as that change was, there was another one that has, in many ways, been just as important: My adoption of the UNIX standard.

THE ADVANTAGE OF STANDARDS

Like anyone that enjoys their work, I try to keep tabs on what is going on in the industry. For me this includes looking at the endless stream of new computer devices both big and small and seeing what kind of hardware and software they use.

One of the things that has become very apparent is that whatever the computing platform, be it a tiny computer no bigger than an Ethernet connector that serves up web pages, the latest smart-phone or e-book reader, or a building-size server cluster, there is one thing they nearly all have in common: Some level of compatibility with UNIX.

Linux and its offshoots (e.g. Android) are probably the best known examples. Various flavours of the Berkley Standard Distribution (BSD) are also quite common, especially when you include derivations like the open-source Darwin OS and its ancestors that form the core of Apple's OS-X operating systems. Even Microsoft provides UNIX compatible subsystems for its Windows operating systems.

The bottom line is that 40 years after it was born, the Application Programming Interface (API) spawned by UNIX has become an almost universal standard. Any programmer who has ever written a command-line application has at least some familiarity with the stdio and stdlib libraries. And if they have written a network server they know what sockets are and have almost certainly used the standard BSD socket API. So when I started writing my own OS, it only made sense that it too should be based on such a widely supported and familiar model.

Adopting a proven, widely supported standard can have many advantages over creating a proprietary interface. One that was critical to my own success was the availability of some very well written textbooks on subjects that were at the core of what I was trying to achieve. For my work on DEBROS, there were two in particular that I referred to constantly. The first is "UNIX Network Programming", by W. Richard Stevens [2]. It has since been expanded to multiple volumes and continues to be highly regarded. The second is "Understanding the Linux Kernel", by Daniel P. Bovet and Marco Cesati [3]. While the latter is a detailed look at the Linux kernel in particular, the topics and issues covered apply to any multitasking kernel. Both books are well written and I highly recommend them for anyone wanting to learn more about network programming or multitasking in general.

For programmers, the use of the UNIX standard means that writing code for DEBROS is no more difficult than writing code for any UNIX compatible OS. While DEBROS will never be fully UNIX compliant, the functions I have implemented use the same parameters and provide the same behaviour whenever practical. So learning to program for DEBROS does not require learning a new API. As proven by another person in our lab, it is possible to compile and test code written for DEBROS on other operating systems like Linux with few if any changes. Probably the most significant difference is simply learning to think on a smaller scale.

KEY LESSONS LEARNED

The primary purpose of an operating system kernel is to manage a set of shared resources. And probably the most important resource that it manages is CPU time.

In the process of writing DEBROS I learned a lot about the concepts and methods that make it possible for a kernel to manage resources effectively. Even the briefest discussion of them all would not fit in this paper. But there is one that, probably more than any of the others, serves to demonstrate both the complexity of the issues an efficient kernel must address and the ingenuity that has gone in to addressing them.

This concept is blocking vs non-blocking function calls. Anyone who has used the UNIX poll() or select() functions has been exposed to this concept to some degree. But what I found most surprising is that it applies to so many other functions, including ones I have used for decades, and yet I was barely aware of it or its importance.

Blocking and the Scheduler

The scheduler is the part of the kernel that is responsible for deciding which task should get the CPU next. While scheduling methods vary widely, they share a common goal: Don't waste CPU cycles.

The most basic piece of information a scheduler uses when choosing a task is what "state" the task is in. There are many different states a task can be in, but the key ones relevant to this discussion can be summarized as "Ready", and "Blocked".

A task is "Ready" if it currently has the CPU or has been pre-empted. The only thing a "Ready" task is waiting for is the CPU.

A task may become "Blocked" when it calls a function to perform an operation and the operation cannot be completed immediately (e.g. writing to a socket). The blocking function initiates the requested operation and then loops, waiting until the operation completes. The simplest possible wait loop looks something like this:

```
while (! eventOccured) { }
```

The problem with this very simple approach is that "blocked" tasks continue to use as much CPU time as the scheduler will give them. Blocking is very common, so this approach wastes large amounts of CPU time.

We can reduce the waste quite a bit with one small change:

```
while (! eventOccured())  { schedule(); }
```

Calling the scheduler directly allows a task to voluntarily give up the CPU. The affect here is to limit the loop to one execution each time the task is resumed.

A significant improvement, but there is still a lot of CPU time spent checking on events that haven't occurred yet. Having the scheduler choose blocked tasks less often, will minimize the waste, but at the cost of increasing the average response time to events.

The ideal situation would be for blocked tasks to use no CPU time at all. Once blocked, a process should not be given the CPU until the event that will allow it to unblock occurs. While not perfect, there is a method that does approach this ideal.

Wait Queues

A wait queue is a doubly-linked list of entities that all have something in common. The most typical use is to keep track of which tasks are waiting for an event related to a shared resource. The wait queue is created by the driver that is responsible for the resource. Tasks are added to the queue by blocking functions when they need to use the resource, and are removed by the driver when the resource becomes available.

The following shows how wait queues can be used in wait loops to make blocking much more efficient:

```
for (;;)  {
  set state to blocked and add to wait queue;
  if (eventOccured)  break;
  schedule();
}
set state to Ready and remove from wait queue;
```

While it may seem a bit odd, the order shown for these steps is critical. Any other order can result in waste we are trying to avoid or, worse yet, a hung task.

The first step is to set the state of the task to Blocked and add the task to the event's wait queue. Note that pre-empted tasks are treated as being in the Ready state, regardless of what their state is set to, so even if the task gets pre-empted after changing its state, it will not hang.

The next step is to check to see if the event has occurred. If it has, we exit the loop.

If the event has not occurred, we call the scheduler for the reasons explained previously. When we call the scheduler, one of two things will be true.

The first is that the task was not pre-empted between when it checked on the event and when it called the scheduler, or it was but the event didn't occur before it was resumed. In that case the task is still in the event's wait queue and its state is still set to Blocked. Because the scheduler was reached by the task calling it directly, it will not be resumed until the event occurs, at which point the driver will remove it from its wait queue and set the task's state back to Ready.

The other possibility is that the task was pre-empted after the check and the event occurred before it was resumed. In that case, the driver will have removed the task from its wait queue and changed its state back to Ready.

One of the subtle aspects of this logic (which was actually missed in an early version of Linux) was that you have to check on the state of the event after adding the task to the wait queue. Checking before you do so doesn't cause a problem, but checking after is critical. Consider what could happen if the only check is before:

- The check returns false and the task is pre-empted before it can add itself to the wait queue.
- The event occurs while the task is pre-empted.
- The task resumes and adds itself to the wait queue.
- The task is now dormant waiting for an event that has already occurred.

Even if the task does get resumed by a later event, at the very least it will have been delayed. Worse, it will have missed the previous event which, depending on the circumstances, could have fatal side effects.

When properly implemented and used by application code, blocking can be a very powerful tool in the quest to create an efficient system. As shown here, not only can a task block indefinitely with little or no overhead, but it will be resumed with minimal delay when the event it was waiting for occurs.

Until I wrote my own OS, the concept of blocking vs non-blocking calls barely penetrated my consciousness. Even when using calls like poll() and select() it never occurred to me how key they are to the efficient operation of the OS, as well as to writing efficient applications.

SUMMARY

The lack of a standard software platform for 8-bit microcontrollers is an impediment to their use in projects they are otherwise well suited for. Proprietary tools and programming interfaces require longer learning curves and result in non-portable software which increases the cost of using them.

DEBROS provides a way to get the benefits of a familiar standards-based programming API and runtime environment on inexpensive hardware, making them a practical alternative in many cases.

REFERENCES

[1] Andrew S. Tanenbaum, "Operating Systems, Design and Implementation," Prentice-Hall, Inc. (1987).

[2] W. Richard Stevens, "UNIX Network Programming", Prentice-Hall, Inc (1990).

[3] Daniel P. Bovet and Marco Cesati, "Understanding the Linux Kernel", O'Reilly Media, Inc. (2006)

A COMBINED ON-LINE ACOUSTIC FLOWMETER AND FLUOROCARBON COOLANT MIXTURE ANALYZER FOR THE ATLAS SILICON TRACKER*

R. Bates, A. Bitadze#, Dept. Physics & Astronomy, University of Glasgow, G12 8QQ, Scotland, UK
M. Battistin, S. Berry, P. Bonneau, J. Botelho-Direito, B. Di Girolamo, J. Godlewski,
E. Perez-Rodriguez, L. Zwalinski, CERN, 1211 Geneva 23, Switzerland
N. Bousson, G. Hallewell, M. Mathieu, A. Rozanov, CPPM, F-13288 Marseille, France
G. Boyd, Dept. Physics & Astronomy, University of Oklahoma, Norman, OK 73019, USA
M. Doubek, V. Vacek, M. Vitek, Czech Technical University, 16607 Prague 6, Czech Republic
K. Egorov, Physics Department, Indiana University, Bloomington, IN 47405, USA
S. Katunin, B.P. Konstantinov Petersburg Nuclear Physics Institute, 188300 St. Petersburg, Russia
S. McMahon, STFC Rutherford Appleton Laboratory, Chilton, Didcot OX11 OQX, UK
K. Nagai, Graduate School of Pure and Applied Sciences, University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8577, Japan

Abstract

An upgrade to the ATLAS silicon tracker cooling control system may require a change from C_3F_8 (octafluoro-propane) to a blend containing 10-30% of C_2F_6 (hexafluoro-ethane) to reduce the evaporation temperature and better protect the silicon from cumulative radiation damage with increasing LHC luminosity.

Central to this upgrade is a new acoustic instrument for the real-time measurement of the C_3F_8/C_2F_6 mixture ratio and flow. The instrument and its Supervisory, Control and Data Acquisition (SCADA) software are described in this paper.

The instrument has demonstrated a resolution of 3.10^{-3} for C_3F_8/C_2F_6 mixtures with ~20%C_2F_6, and flow resolution of 2% of full scale for mass flows up to $30gs^{-1}$. In mixtures of widely-differing molecular weight (mw), higher mixture precision is possible: a sensitivity of < 5.10^{-4} to leaks of C_3F_8 into the ATLAS pixel detector nitrogen envelope (mw difference 160) has been seen.

The instrument has many potential applications, including the analysis of mixtures of hydrocarbons, vapours for semi-conductor manufacture and anaesthesia.

INTRODUCTION

An upgrade to the ATLAS silicon tracker cooling control system may require a change from the present C_3F_8 evaporant (molecular weight = 188) coolant [1] to a blend with 10-30% of the more volatile C_2F_6 (mw = 138). Central to this upgrade a new acoustic instrument for real-time measurement of C_3F_8/C_2F_6 mixture ratio and flow has been developed, exploiting the phenomenon that the sound velocity in a binary gas mixture at known temperature and pressure depends solely on the molar concentrations of its components. This instrument builds upon the technology of ultrasonic gas analysis used in Cherenkov radiation detectors since the 1980s [2] and measures the molar concentrations of the two fluorocarbon components in the recirculating exhaust vapour following the evaporative cooling of the silicon tracker.

In the custom electronics sound bursts are sent via ultrasonic transceivers◊ parallel and anti-parallel to the gas flow. A fast transit clock is started synchronously with burst transmission and stopped by over-threshold received sound pulses. Rolling average transit times in both directions, together with temperature and pressure, enter a FIFO memory and are passed to a supervisory computer via RS232 or CANbus.

Gas mixture is continuously analyzed using SCADA software implemented in PVSS-II [3], by comparing the average sound velocity in both directions with stored *velocity vs. concentration* look-up tables. These tables may be created from prior measurements in calibration mixtures or from theoretical thermodynamic calculations. Flow rates are calculated from the difference in transit time in the two directions. In future versions these calculations may be made in an on-board microcontroller.

Within the ATLAS experiment the instrument has been used for flowmetry and mixture analysis of C_3F_8/C_2F_6 blends and also as a sensitive detector of leaks of the present C_3F_8 evaporative coolant into the ATLAS pixel detector nitrogen envelope.

MECHANICS

The mechanical envelope and ultrasonic transducer mounting are illustrated in Fig. 1. The transducers are mounted around 660 mm apart in a flanged stainless steel tube of overall length 835mm. The temperature in the tube is monitored by six NTC thermistors – (100kΩ at 25ºC) - giving an average temperature measurement uncertainty of better than ±0.3C. Pressure is monitored with a transducer having a precision better than ±15mbar.

* We acknowledge CERN and home institute support of this project
# a.bitadze@physics.gla.ac.uk

◊ Model 600 50kHz instrument grade ultrasonic transducer: SensComp, Inc. 36704 Commerce Rd. Livonia, MI 48150, USA

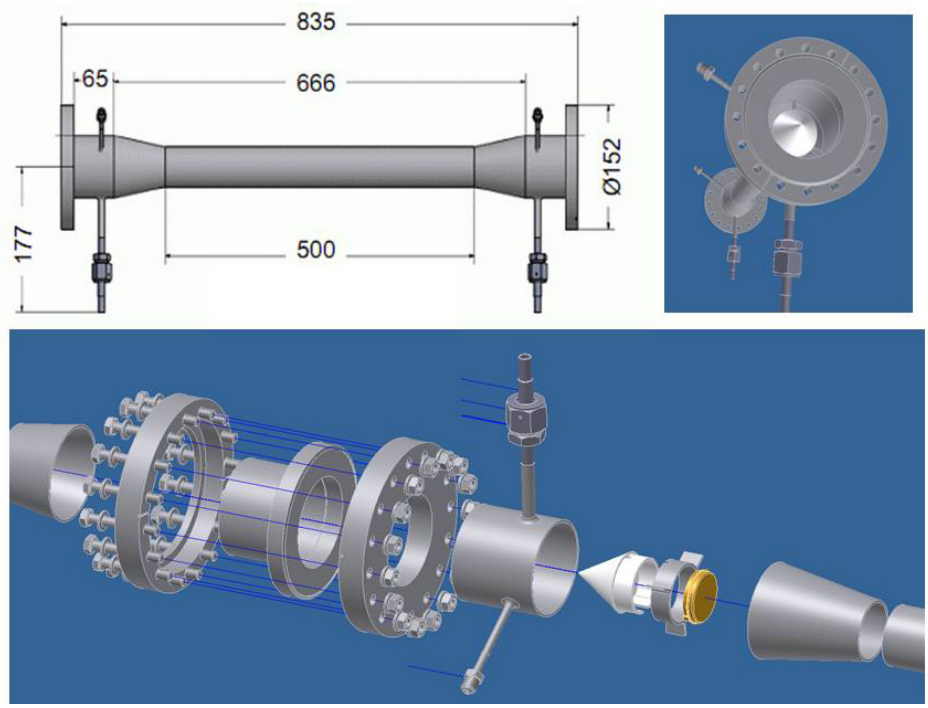


Figure 1: Views of the instrument mechanical envelope, showing an ultrasonic transducer, its mounting and axial flow-deflecting cone, together with tubes for pressure sensing and the evacuation and the injection of calibration gas.

ELECTRONICS

The custom electronics is based on a Microchip ® dsPIC 16 bit microcontroller. This generates the 50kHz sound burst signals emitted by the transducers and includes a 40 MHz transit clock that is stopped when an amplified sound signal from a receiving transducer crosses a user-definable comparator threshold.

The HV bias for the vibrating foils of the capacitative transducers, settable in the HV range 180-360V, is generated by a DC-DC converter. When transmitting, a transducer is excited with a train of (1-8) HV square wave pulses, built using the 50kHz LV pulses from the microcontroller and the DC-DC converter output. When receiving, a transducer is biased with a flat HVDC bias and its signal passed to an AD620N amplifier followed by a comparator.

Sound transit times are computed from the number of 40MHz pulses counted between the rising edge of the first transmitted 50KHz sound pulse, and the time the first received, amplified sound pulse crosses the comparator threshold. Transit times, computed alternately in the two transmission directions are continuously entered into an internal FIFO memory. When a measuring cycle is requested by the supervisory computer a time-stamped running average from the 300 most recent transit times in each direction in the FIFO memory is output, together with the average temperature and pressure, at a rate of up to 20 averaged samples per second.

In addition to the I/O connectivity for communications, the ultrasonic transducers, pressure and temperature sensors, two (4-20 mA) analog outputs provide feedback for adjustment of the C_3F_8/C_2F_6 mixing ratio by the external gas mixture control system.

CALIBRATION AND MEASUREMENTS IN PURE C_2F_6 AND C_3F_8

For high precision mixture and flow analysis the uncertainty in the sound flight distance should be minimized. It is necessary to perform a one-time transducer foil inter-distance calibration. The most convenient method is to calculate this distance from an average of measured sound transit times with the tube filled with a pure gas (or gases) having well-known sound velocity dependence on temperature and pressure. We initially made calibrations using xenon, whose sound velocity and mw (175.5 ms^{-1} at 20ºC, 137 units) are closest [4], [5] to those of the fluorocarbon mixtures in the ATLAS application [1], and whose thermo-physical behaviour is that of an ideal gas. Later calibrations demonstrated sufficient precision with nitrogen and argon, which are considerably cheaper and more widely available. The average uncertainty in transducer inter-distance measured in this way is ± 0.1mm.

Multiple measurements were made in pure C_3F_8 and C_2F_6 at around 19.5°C, with pressure in the range 0.4 - 2.7bar_{abs}[6]; conditions expected in an acoustic vapour analyzer installed in the (superheated) vapour return path some tens of metres from the evaporative zone within the ATLAS silicon tracker. The average difference between measured sound velocities and the predictions from a PC-

SAFT[□] equation of state (EOS) [6], [7], [8]) was less than 0.04% in both fluids.

SCADA & ANALYSIS SOFTWARE

The specialized software for the gas analyzer operation is coded as a standalone component in the PVSS II, v3.8 SCADA environment [3], [6]; a standard at CERN. Its main tasks include:

- vapour flow rate determination;
- sound velocity and molar vapour mixture concentration determination;
- RS232 or CANOpen communication, to start and stop the measuring cycle, and to request time-stamped bidirectional sound transit times, temperature and pressure data from the instrument FIFO memory;
- calculation and transmission of the set-points for the analog (4-20mA) output signals for C_3F_8/C_2F_6 ratio adjustment in the external cooling plant;
- visualization via a Graphical User Interface (GUI);
- archiving of sound transit times, velocities, flow, mixture composition, temperature and pressure into a local and/or remote data base.

The vapour flow rate is calculated from the sound transit times measured parallel, t_{down}, and anti-parallel, t_{up}, to the flow direction, according to the following algorithm:

$$t_{down} = L / (c + v), \quad t_{up} = L / (c - v) \qquad (1)$$

where v is the linear flow velocity (ms^{-1}), c the speed of sound in the gas and L the distance between transducers.

The gas volume flow V (m^3s^{-1}) can therefore be inferred from the two transit times by:

$$V = L/2 * A * ((t_{up} - t_{down})/ t_{up} * t_{down}) \qquad (2)$$

where A is the internal cross sectional area of the axial flow tube between the two ultrasonic transducers (m^2).

The sound velocity c can also be inferred from the two transit times via:

$$c = L/2 * ((t_{up} + t_{down})/ t_{up} * t_{down}) \qquad (3)$$

It can be seen from Eqs. (1) – (3) that knowledge of the temperature of the gas is not necessary for flowmetry.

Figure 2 shows the linearity of the ultrasonic flowmeter element of the instrument in C_3F_8 vapour at 20°C through comparison with a Schlumberger Delta G16 gas meter, at flows up to 230 $lmin^{-1}$ (~30 gms^{-1}); the maximum mass flow in the presently-available C_3F_8/C_2F_6 blend circulation system. The average precision is 2% of full scale.

The calculation of gas mixture molar ratio requires the use of ("*c, t, p*") look-up tables of gas mixture composition in the binary mixture to be analyzed, corresponding to sound velocities, *c*, at known temperatures, *t,* and pressures, *p*).

Look-up table data may be gathered from prior measurements in calibration mixtures or from theoretical data. Fig. 3 compares measured sound velocities in calibrated mixtures of C_3F_8 and C_2F_6 with sound velocity predictions at 19.2°C from a PC-SAFT EOS [7], [8] and the refrigerant-oriented extended Benedict-Webb-Rubin

[□]Purturbed-Chain Statistical Associating Fluid Theory

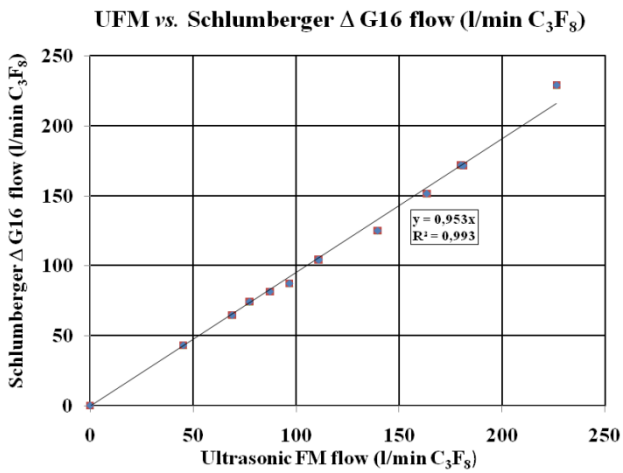


Figure 2: Ultrasonic flowmeter linearity comparison with a Schlumberger Delta G16 gas meter: C_3F_8 vapour.

(BWR) EOS used in the NIST REFPROP thermodynamic software package [9]. The (0→35%) C_2F_6 concentration range spans the region of thermodynamic interest to the ATLAS silicon tracker cooling application.

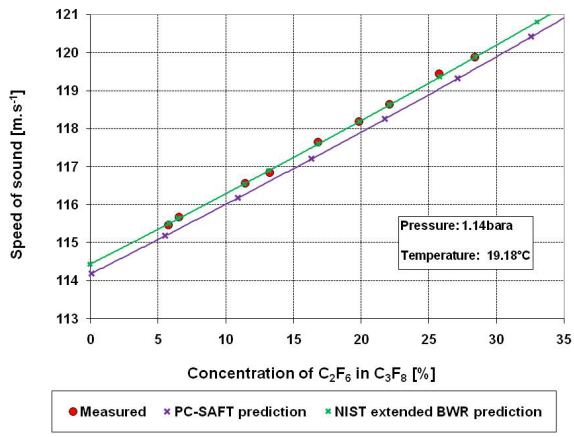


Figure 3: Comparison between measured sound velocity data and theoretical predictions in C_3F_8 / C_2F_6 mixtures.

The mixtures of C_3F_8 and C_2F_6 shown in Fig. 3 were set up by partial pressure ratio in the previously-evacuated tube, creating a molar ratio binary gas mixture. The transducer foil inter-distance had been previously established using the above-described gas calibration procedure, to a precision of ± 0.1mm.

The average difference between measured and the PC-SAFT and NIST-REFPROP predicted sound velocities in mixtures with (0→35%) C_2F_6 in C_3F_8 were respectively 0.5% and 0.05% at pressures around 1 bar_{abs} and temperatures in the range 15-25°C. It is recognised that the present version of the NIST-REFPROP [5] database is the more precise in predicting the thermophysical properties of mixtures of saturated fluorocarbons (having molecular structures of the form $C_nF_{(2n+2)}$).

The precision of mixture determination, $\delta(mix)$, at any concentration of the two components is given by;

$$\delta(mix) = \delta c/m \qquad (4)$$

where m is the local slope of the sound velocity/ concentration curve and δc is the uncertainty in the sound velocity measurement - dependent on transit time resolution, transducer spacing and uncertainties in the measured temperature and pressure (±0.2°C, ±5mbar in this instrument) or variations between these parameters and the (t, p) values of the nearest (c, t, p) curve in the calibration database. For example, at a sound velocity of ~118 ms^{-1}- corresponding to a blend of 20% C_2F_6 in C_3F_8 (Fig. 3) - the combined measurement uncertainties result in a sound velocity uncertainty of 0.06 ms^{-1}, yielding a concentration uncertainty ~0.3% at 20%C_2F_6, where the slope of the velocity/ concentration curve is ~0.18ms^{-1}%$^{-1}$.

The present software [6] uses a pre-loaded look-up table of NIST-REFPROP BWR-generated sound velocity with 0.25% granularity in C_3F_8/C_2F_6 molar mixture and covering the expected range temperature and pressures (16.2→26.1°C, 800→1600$mbar_{abs}$ with 0.3°C & 50mbar granularity). The algorithm calculates mixture composition by minimizing a quadratic norm, n_i, for each (c_i, T_i, P_i) table entry:

$$n_i = k_1(p_{i,\,table} - p_{running\,average})^2 + k_2(t_{i,\,table} - t_{running\,average})^2 + k_3(c_{i,\,table} - c_{running\,average}) \quad (5)$$

where $k_{1,2,3}$ are sensitivity parameters [6] and p, t & $c_{running\ average}$ are real-time outputs of the instrument FIFO memory.

A new software version [6] will implement a database covering a much larger c, T, P range and will allow "zooming" to smaller sub-tables (O~10,000 c, T, P data points), corresponding to a narrower process range - as in the present application.

In a second application related to the ATLAS evaporative cooling system, we use ultrasonic binary gas analysis to detect low level C_3F_8 vapour leaks into the N_2 environmental gas surrounding the ATLAS silicon tracker. Figure 4 compares measured sound velocities in mixtures containing up to 10% C_3F_8 in N_2 with sound velocity predictions from the PC-SAFT EOS[9]. A reduction in sound velocity of 0.6 m/s from the base velocity of ~351 ms^{-1} was seen during a long term (> 1 year study). From the ~-12.27ms^{-1}%$^{-1}$ average gradient of the sound velocity-concentration curve at trace C_3F_8 concentrations in the range 0→0.5% (Fig. 4) this sound velocity indicated, using Eq.(4), a C_3F_8 leak ingress of 0.049%, later traced to one of 204 evaporative cooling circuits into the ATLAS silicon tracker nitrogen envelope.

CONCLUSION

We have developed a combined, real-time ultrasonic flowmeter and binary gas analyzer, whose accuracy was determined following calibration in pure reference gases - by a set of measurements in C_3F_8/C_2F_6 blends. Sound velocity measurements were within 0.05% of the predictions of the NIST-REFPROP package, allowing mixture resolution of 0.3% in the (0-35% C_2F_6) concentration range of interest for the ATLAS silicon tracker.

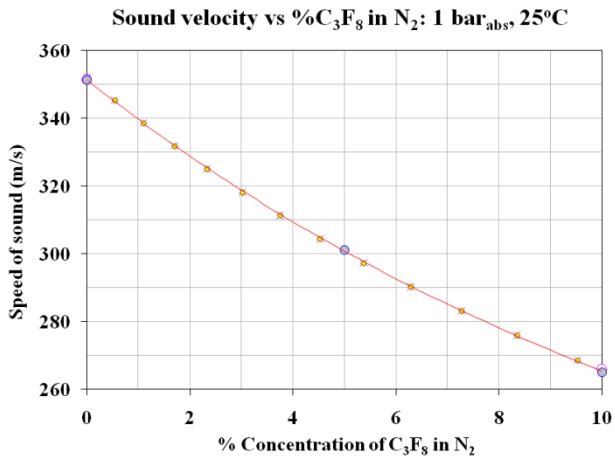


Figure 4: Comparison of sound velocity measurements (°) and PC-SAFT predictions (◊) in C_3F_8/N_2 mixtures.

The instrument presently analyzes vapour mixtures of C_2F_6/C_3F_8 and N_2/C_3F_8, respectively having a molecular weight difference of 50 and 160 units, the mixture resolution being seen to increase with the mw difference of the components. The instrument has applications in the analysis of hydrocarbon-air mixtures, refrigerant-air mixtures (leak detection), vapour mixtures for MOCVD semiconductor manufacture and anaesthetic gas mixtures.

REFERENCES

[1] D. Attree et al (96 authors), "The evaporative cooling system for the ATLAS inner detector." JINST 3:P07003 (2008) 1

[2] G. Hallewell, G. Crawford, D. McShurley, G. Oxoby and R. Reif, "A sonar-based instrument for the ratiometric determination of binary gas mixtures", Nucl. Instr. & Meth A 264 (1988) 219

[3] PVSS II – Process visualization and control system, Version 3.8 (2009) ETM professional control GmbH, A-7000, Eisenstadt, Austria http://www.etm.at

[4] V. Vacek, G. Hallewell and S. Lindsay, "Velocity of sound measurements in gaseous per-fluorocarbons and their mixtures", *Fluid Phase Equilibria*, 185(2001) 305

[5] V. Vacek, G. Hallewell, S. Ilie and S. Lindsay, "Perfluorocarbons and their use in cooling systems for semiconductor particle detectors", *Fluid Phase Equilibria*, 174(2000) 191

[6] R. Bates et al (21authors), "An on-line acoustic fluorocarbon coolant mixture analyzer for the ATLAS silicon tracker" Proc. 2nd International conference on Advancements in Nuclear Instrumentation, Measurement Methods and their Applications (ANIMMA) Ghent, Belgium 6-9 June, 2011

[7] G. Hallewell, V. Vacek and V. Vins, "Properties of saturated fluorocarbons: Experimental data and modeling using perturbed-chain-SAFT" , *Fluid Phase Equilibria* 292(1-2): 64-70 (2010)

[8] G. Hallewell, V. Vacek and M. Doubek, "Novel and simple sonar gas analyzers", *Proc 9th Asian Thermophysical Properties Conference*, Beijing, October 19–22, CD ROM Paper No.:109296: 1-6 (2010)

[9] E. Lemmon, M. Huber and M. McLinden, 'REFPROP' Standard reference database 23, version 9.0 U.S. National Institute of Standards and Technology (2010)

INTERCONNECTION TEST FRAMEWORK FOR THE CMS LEVEL-1 TRIGGER SYSTEM

J. Hammer, CERN, Geneva, Switzerland
M. Magrans de Abril, Wisconsin University, Madison, Wisconsin, U.S.A.
C-E. Wulz, Austrian Academy of Sciences, Vienna, Austria

Abstract

The Level-1 Trigger Control and Monitoring System is a software package designed to configure, monitor and test the Level-1 Trigger System of the Compact Muon Solenoid (CMS) experiment at CERN's Large Hadron Collider. It is a large and distributed system that runs over 50 PCs and controls about 200 hardware units.

The objective of this paper is to describe and evaluate the architecture of a distributed testing framework – the Interconnection Test Framework (ITF). This generic and highly flexible framework for creating and executing hardware tests within the Level-1 Trigger environment is meant to automate testing of the 13 major subsystems interconnected with more than 1000 links. Features include a web interface to create and execute tests, modeling using finite state machines, dependency management, automatic configuration, and loops. Furthermore, the ITF will replace the existing heterogeneous testing procedures and help reducing both maintenance and complexity of operation tasks.

INTRODUCTION

The Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC) of CERN, the European Organization for Nuclear Research, is a detector designed to find answers to some of the fundamental open questions in physics today [1].

Since millions of collisions of protons occur every second and only a small fraction of these can provide insight into new physics, events have to be selected on-line according to their properties. This is done by the trigger system, a vital part of the experiment.

The trigger system is organized in two levels: (1) The Level-1 Trigger (L1T) [2] is a custom-designed, largely programmable electronic system which preselects the most interesting collisions for further evaluation. These collisions are then examined and – if selected – stored permanently by (2) the High-Level Trigger (HLT) [3], which consists of a farm of industrial processors.

The Trigger Supervisor Framework

The L1T is configured, tested and monitored using the Level-1 Trigger Control and Monitoring System – a large and distributed system that runs over 50 PCs and controls about 200 hardware units (~ 6000 boards). Each of the components of this system is based on the Trigger Supervisor (TS) Framework [4], written in C++, providing a web interface utilizing AJAX.

The TS architecture is composed of a hierarchical tree of nodes, where the central (i.e. the top) node is in charge of coordinating the access to the subsystems [5]. Each node is accessible by a well-defined interface based on the Simple Object Access Protocol (SOAP) [6], and can run one or more commands and operations simultaneously. Operations are stateful objects that use transitions to move between different states of their internal finite state machine (FSM) – which can be defined freely.

The communication hierarchy is strictly top-down – following the request-response model – with subsystems not even knowing their supervisor nodes. Fig. 1 shows an example system setup.

Need for an Interconnection Test Framework

With more than 1000 links connecting the about 200 hardware units, thorough testing is crucial. So far, the various subsystems have used custom scripts for testing – controlling the TS nodes via SOAP commands. Needless to say, this leads to a lot of duplicated effort with a high level of maintenance required. Even more important, this approach does not allow the centrally controlled execution of tests (in particular by non-experts!), nor to execute them automatically at specific events.

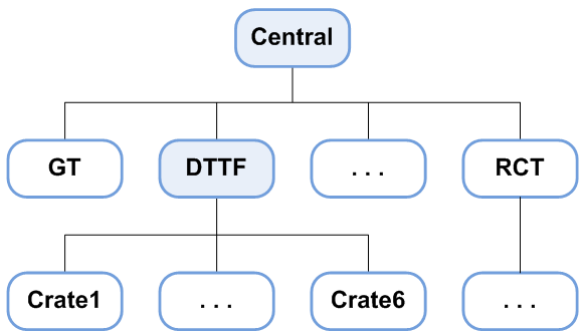


Figure 1: An example Level-1 Trigger system setup.

Instead, local test methodologies heavily rely on the respective subsystem experts and time-consuming coordination between them and the global run control operators. All these factors lead to unnecessary system downtime and a waste of precious manpower, both undesirable consequences for such a huge experiment involving large resources of manpower and investment.

A much better solution would be to provide an interconnection test framework on top of the existing TS framework. This could make use of the already available infrastructure and facilities, and provide additional common functionality for distributed testing. However, although the basic idea was already stated in [6], a sophisticated implementation was yet to be developed.

Requirements and Constraints

Among the requirements:

- Main focus on simplicity: The users should need as little effort as necessary to use the ITF (simple API, little setup/configuration, …).
- Tests can be run both locally and centrally controlled.
- Tests can be run in different network environments (with or without database; with various and even fake top nodes, …).
- Unlimited subsystem hierarchy depth.
- Shall be very flexible to allow a large range of diverse tests (including self-tests).

Imposed constraints were:

- Shall be embedded into the existing Trigger Supervisor environment, i.e. the tests shall be implemented using TS operations.
- Shall allow both interactive and scripted execution.

ARCHITECTURE

Overview

The Interconnection Test Framework is built on top of the Trigger Supervisor Framework, and therefore fully embedded into the existing environment. As a result, the ITF is highly TS specific, using the available API to communicate with subsystems and drive operations. Figure 2 shows the basic architecture:

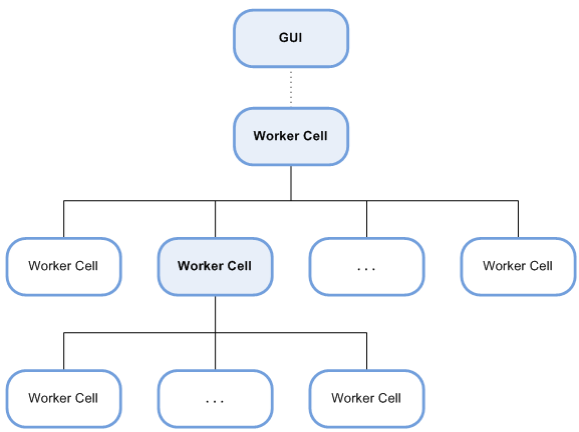


Figure 2: The architecture of the Interconnection Test Framework.

Each node runs one Interconnection Test operation – a sub class of the framework's base class, with the concrete functionality implemented by the user. Any node will automatically act as a supervisor once there are sub nodes defined for it.

Which and how many nodes (i.e. systems) are involved depends on the test case – essentially an XML document – that defines all the necessary details for a specific test (e.g. user-defined parameters).

Service Discovery (ITF-SD)

Much effort was spent on simplicity. Wherever possible, the "convention over configuration" paradigm was applied to reduce both the required initial effort to create a test case as well as the necessary maintenance.

One core feature that facilitates this strategy is the Service Discovery (ITF-SD). It allows to automatically detect all available test operations in the network which are reachable from the start node. Based on this information is automatically creates test cases without having to define the sub nodes – exploiting the already defined system hierarchy.

This eliminates the need to maintain the test cases when the topology of the network (or just the name of a subsystem) changes. Both partly pre-defined (auto-detecting a specific sub tree only) and on-the-fly test cases (auto-detecting the entire test tree) are possible.

The Finite State Machine (FSM)

The finite state machine is the main building block of a test operation. It defines the available system states and the legal transitions to move between them, and serves as the common specification for all test operations. Consequently, the FSM should be quite flexible to allow a diverse set of test types – starting from self-tests that would need only one single transition, to complex distributed tests that require loops with plenty of synchronization between the nodes.

Figure 3 shows the finite state machine that is used for the Interconnection Test Framework. It features six transitions: setup, prepare, execute, analyze, next and summarize. Each test has at least one loop, the number being either defined statically in the test case definition or set dynamically at runtime (by any node).

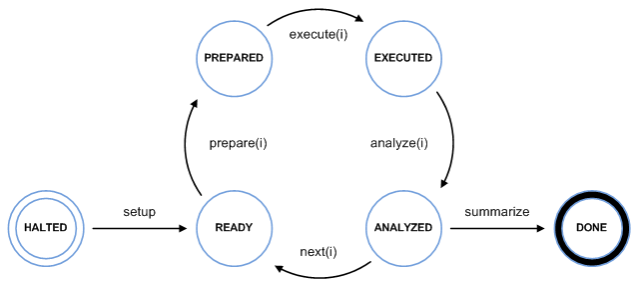


Figure 3: The finite state machine for all test operations.

FSM Pruning

As shown in the previous section, the FSM features a loop. Each loop provides four different transitions – prepare, execute, analyze and next. Each of these transitions has to be called in every subsystem in every loop. However, most test operations will not need that many steps, e.g. execute and analyze only might be sufficient. Nevertheless, the empty transitions have to be called as well – leading to a significant performance penalty when many loops are required (not to mention that every connection to be made has a risk to fail).

To get rid of these disadvantages while still providing a flexible and unique FSM, FSM pruning (or transition-skipping) is applied. During the first loop the framework automatically detects which transitions are empty (i.e. not in use) and uses this information to skip those transitions

in the following loops. This works on a per sub-tree basis. The only thing the developer has to do is not to implement the corresponding transition.

Figure 4 shows the actual FSM that will be utilized in case the prepare transition is empty.

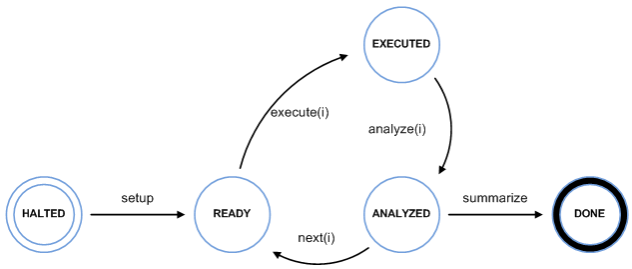


Figure 4: FSM Pruning: Skipping the prepare transition.

Sequences

Sequences are used to define the order in which a transition should be executed in different subsystems of the same level of the hierarchy. Their main purpose is to define dependencies. If nothing is specified explicitly, the default rule is to execute everything in parallel.

Shared Memory

The communication in the Trigger Supervisor system is strictly top-down. However, for some tests it is convenient or even necessary to communicate with other (e.g. the neighbor) nodes.

Therefore, the ITF provides a simple shared memory, based on name/value pairs. A name may also refer to a file on the local hard disk – the framework will automatically take care of syncing it to any node that needs the file.

The Test Result

The basic building block of a test result is the ResultItem. A ResultItem – either of type success or failure – may contain a message and additional attributes (which allows structured information). Neutral items (so-called properties) are possible too to include general information, e.g. the current state of the hardware.

It is also possible to store detailed result data in separate files. A test may generate as many ResultItems as required. To issue a failure with an empty message and two attributes all that is required is something like

```
failure("").attributes()
      .add("source", boardID)
      .add("expected", pattern));
```

(Custom) Views / Analyzers

In order to visualize the test result in a user-friendly way, result views are provided. With the default view (Fig. 5) the ResultItems of all systems and loops can easily be examined. By using AJAX even huge test results with thousands of items can be browsed through without putting a heavy load on the browser.

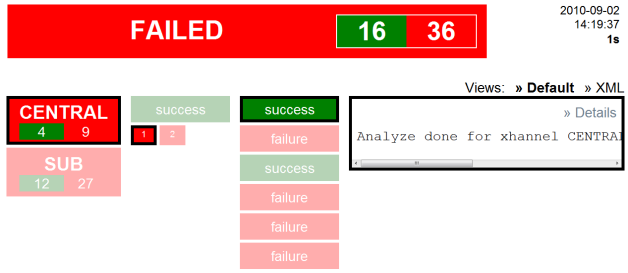


Figure 5: The framework's default result view.

In order to provide higher flexibility and facilitate a wide variety of test types, custom result views / analyzers are possible as well (like the one in Fig. 6). This allows interpreting the result in a specific way and, in particular, visualizing it according to the needs of that type of test.

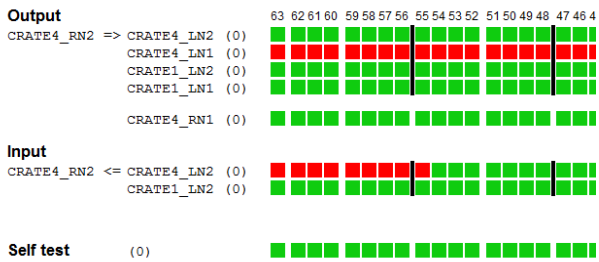


Figure 6: A custom result view.

Central History

All test results are automatically stored into the database to provide a central history. Therefore, the results can be evaluated and compared to previous results at any later time. As one of the goals of the ITF is to be database independent, this will fail silently in case no database is available.

As an additional feature automatic logging of all actions and (sub) results is provided to help debugging issues.

RELATED WORK

The CMS experiment with its many custom-made hardware units is, well, unique. Therefore, the Trigger Supervisor - a system which is responsible for centrally controlling all those units – already had to be developed in-house as there was no other solution available. Thus, the main constraint was that the Interconnection Test Framework must run within the Trigger Supervisor environment.

As a result, there was no other solution than to develop the ITF from scratch for the specific needs of the CMS Level-1 Trigger, after gathering the requirements from different subsystem groups. The basic idea has already been stated in [7], including an original, loop-free version of the finite state machine.

In the wider sense, all the XUnit frameworks can be considered related work. However, although some concepts are similar and ideas were derived in particular

from JUnit [8], those are frameworks for unit testing and not for distributed testing.

CONCLUSION

This paper introduced the Interconnection Test Framework for the CMS Level-1 Trigger System, a solution for testing connections between several nodes of a distributed system. The framework is built on top of the Trigger Supervisor framework, and allows executing transitions with arbitrary actions synchronously on several systems.

The excellent flexibility and extensibility of the Interconnection Test Framework was shown – both a wide variety of test types as well as custom result analyzers are possible. Even more important is that the framework tries to reuse many concepts of the underlying Trigger Supervisor framework, enabling the test developers to start right away by using already familiar knowledge.

However, this also means that the concrete implementation of the ITF is applicable for our environment only. Nevertheless, the concepts and the conclusions are hopefully useful for others as well.

Another limitation is due to the fact that XML is quite verbose, which limits the amount of data that can be stored without affecting performance. With the current implementation, about 10000 result items is a reasonable limit (could be improved if necessary).

Still, XML is an ideal choice for both the configuration and the result as it is highly flexible and easy to maintain (in contrast to e.g. JSON). It makes development so much faster if you are able to add attributes or entire trees on-the-fly without having to change anything in your code.

Simplicity is the main prerequisite for user acceptance, and thus the key ingredient for making the framework successful. This was achieved by reusing the concepts of the TS framework and by keeping the required test configuration to a bare minimum with features like the Service Discovery.

The second critical factor is performance. After some analysis we figured out the minimal Finite State Machine that would be needed to implement our tests. As four steps per loop would slow down many tests, we decided to automatically prune the FSM instead of providing several versions. This makes life easier for the developers but still allows high performance.

Although the framework is constantly being developed further, it already provides a stable environment for various kinds of tests in the CMS Level-1 Trigger system where it proved to be immediately applicable for real-world use cases.

ACKNOWLEDGMENT

We would like to thank Christian Hartl and Thomas Themel for providing ideas and valuable feedback. Franz Mittermayr developed the original versions of the test configuration and execution user interfaces.

REFERENCES

[1] CMS Collaboration, “CMS Technical Proposal”, CERN/LHCC 94-38, 1994.
[2] CMS Collaboration, “The TriDAS Project – The Level-1 Trigger Technical Design Report”, CERN/LHCC 2000-38, 2000.
[3] CMS Collaboration, “The TriDAS Project – Data Acquisition and High-Level Trigger Technical Design Report”, CERN/LHCC 2002-26, 2002.
[4] I. Magrans de Abril, C-E. Wulz, J. Varela, “Concept of the CMS Trigger Supervisor”, IEEE Trans. Nucl. Sci. Vol. 53 Nr. 2, 474-483, 2006.
[5] I. Magrans de Abril and M. Magrans de Abril, “The CMS Trigger Supervisor Project”, IEEE Nuclear Science Symposium Conference Record, Puerto Rico, 23-29 October, 2005.
[6] Simple Object Access Protocol, http://www.w3.org/TR/soap.
[7] I. Magrans de Abril, “The CMS Trigger Supervisor: Control and Hardware Monitoring System of the CMS Level-1 Trigger at CERN”, Doctoral Thesis, Universitat Autonoma de Barcelona, 2008.
[8] The JUnit Framework, http://www.junit.org.

A TESTBED FOR VALIDATING THE LHC CONTROLS SYSTEM CORE BEFORE DEPLOYMENT

J. Nguyen Xuan, V. Baggiolini, CERN, Geneva, Switzerland

Abstract

Since the start-up of the LHC, it is crucial to carefully test core controls components before deploying them operationally. The Testbed of the CERN accelerator controls group was developed for this purpose. It contains different hardware (PPC, i386) running various operating systems (Linux and LynxOS) and core software components running on front-ends, communication middleware and client libraries. The Testbed first executes integration tests to verify that the components delivered by individual teams interoperate, and then system tests, which verify high-level, end-user functionality. It also verifies that different versions of components are compatible, which is vital, because not all parts of the operational LHC control system can be upgraded simultaneously. In addition, the Testbed can be used for performance and stress tests. Internally, the Testbed is driven by Atlassian Bamboo, a Continuous Integration server, which builds and deploys automatically new software versions into the Testbed environment and executes the tests continuously to prevent from software regression. Whenever a test fails, an e-mail is sent to the appropriate persons. The Testbed is part of the official Controls System development process wherein new releases of the controls system have to be validated before being deployed operationally. Integration and system tests are an important complement to the unit tests previously executed in the teams. The Testbed has already caught serious bugs that were not discovered by the unit tests of the individual components.

MOTIVATION

As described in the previous publication [1], the accelerator controls system can roughly be described as 3 tier architecture, mainly written in Java and C/C++. It is composed of many layers developed by separate teams. Along with software, hardware has also been evolving and nowadays the operational environment consists of different hardware architecture running different OS at the same time.

Upgrades are very challenging since many components and teams are involved. There is no place for failure, since a beam downtime of the LHC itself only costs about 50'000CHF/h.

The different components are well tested individually with unit tests, but without any systematic function and integration tests. This is the reason why we started the Testbed project.

THE TESTBED

Overview

The main goal of the Testbed is to test components together before they are deployed into operations and validate a working set of versions. This practice is part of an overall development process to which also the SIP initiative belongs [2].

The scope of the Testbed is to test general purpose controls components. It does not include GUIs, or devices with a specific function, such as a power converter or a beam loss monitor. The tests focus on functional aspects, mainly integration and system testing, but also include verification of backward compatibility and regression testing. Some tests validate the reaction to failures. Failure can be provoked artificially by shutting down services in the Testbed, and that without disturbing the operational environment. The Testbed is composed of several different machines, representing the variety of hardware and operating systems used in the CERN accelerator complex. The core software components of the controls system are deployed on these machines and clients are emulated, a detailed description follows in the next section.

Diving into Details

The Testbed tries to mimic the operational environment of the core components (Fig. 1).

Starting from the bottom, FECs (Front-End Computer) are needed to send simulated data and are synchronized by the timing system. Both hardware types from operations are integrated: PPC with LynxOS and i386 with Linux. On these FECs, two implementations from different generation are used: the PS's GM (General Module) and the LHC's FESA (Front End System Access).

The middle-tier is composed of common middleware services which are:

- the CCDB (Controls Configuration Database) containing data essential for most of the components
- a JMS (Java Messaging System) broker to pass messages
- RBAC (Role Based Access Control) security to restrict some actions to certain users
- CMW (Controls Middleware) services as the directory server which provides the different server addresses and the proxy which protects FECs from too many connections

Finally at the top, communication libraries with APIs like JAPC (Java API for Parameter Control) or RDA (Remote Device Access) commonly used by clients are tested.

This setup allows calls from client APIs with various paths and properties, as fetching a value from a device directly or through the proxy, enabling RBAC authentication, and so on.

THE TESTS

Type of Tests

As already mentioned, the scope of the Testbed is functional and system testing. Tests validate (1) the typical primary functions and interactions the controls system provides, (2) the correct reaction of the controls system to typical errors (e.g. device failures) and (3) the backward compatibility of new components.

An example of primary functionality is device access. All controls devices implement a device/property model. The most basic interactions with such a device are *set*, *get* and *subscription* on the properties. They can be triggered at various API levels, at the lower level RDA API and at the higher level JAPC API. To validate this functionality a test first reads the property value, then sets the property to a new value, and finally checks that the new value is published through the subscription mechanism.

Correct reaction to failures can be tested with a special device class that simulates typical device errors, e.g. sending wrong data or not responding at all. Tests check amongst other things that correct exceptions with the right error messages are thrown.

Backward compatibility is checked by deploying the new and the old version (e.g. of the communication middleware) into the Testbed, and checking that they interoperate correctly. In general, different version combinations that may occur are tested, e.g. different versions of FESA with different versions of the middleware components.

Test Organization

The order in which tests are executed is important to make sure we get accurate test results.

Before running the actual function tests, the Testbed runs a series of self-tests, called preconditions tests. They verify that all Testbed components are active and configured as expected. For example, they check if a device is online, if the timing works, if the directory service and the database are accessible. If any of the preconditions fails, the Testbed stops, waiting for the defective components to be fixed. Otherwise the process continues and the actual function tests are executed. Preconditions tests are important to keep the test results clean and correct; they make sure that functional tests only fail on real malfunctions in the controls system, not because of misconfigured Testbed components.

The set of tests is also run in a well-defined, bottom-up sequence. Because a higher-level test will involve all those previous components, we first make sure that the lowest components are fully operational before running tests on higher components. First the tests targeting low level components are run, such as tests on the timing system, the directory service or the proxy. If a test on the proxy fails, then tests from JAPC are likely to also fail. By testing the proxy before JAPC, we ensure that a fault in the proxy is recognized as such, and not as a JAPC fault.

Writing and Maintaining Tests

Tests should be written by the teams who provide the library or the component or by a team dedicated to testing in order to be able to keep up with the project changes or to simply maintain the tests. But in practice the teams focus more on development of new features, so the tests are often written by third persons following the specifications.

The Testbed administrator is in charge of verifying and integrating new tests into the test suite. It happens that bad written tests give wrong errors result, but they are fairly easy to spot and quickly fixable. Those errors are

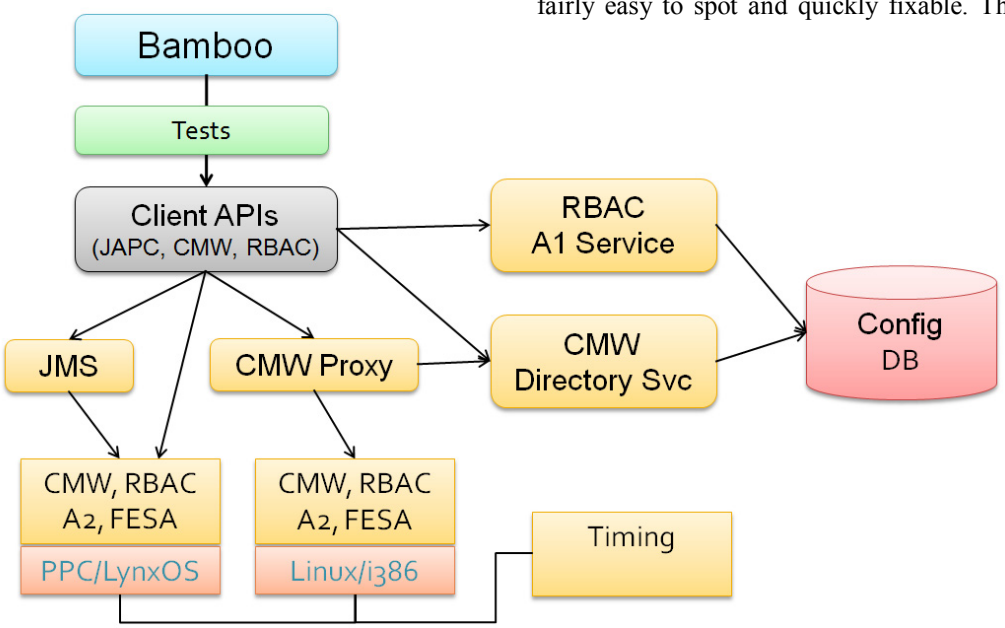


Figure 1: The Testbed structure.

habitually detected at development time.

Tests are updated when a new component with non backward compatible changes are deployed. Usually the new deployed components make the tests fail, so unless the tests are changed, they will keep failing.

Running Tests

Tests are run by our continuous integration server Atlassian Bamboo [3]. Normally, Bamboo is used as follows. A so-called Bamboo test plan monitors a source repository and triggers a build process when source code changes are committed by some developer. A build process checks out the source code, compiles the sources and runs the unit tests. If all this succeeds, a cascade of other dependent test plans can be triggered. If it fails, Bamboo sends typically an e-mail to the owner of the test plan and to the committer. Bamboo displays the results of unit tests very nicely with graphs, relevant statistics and metrics, and the detailed logs of the whole execution are available. A history of all test plan executions is kept as well. Within a test plan, several stages can be defined to divide a build into several steps. These steps are run sequentially and we use them to run the tests in a defined bottom-up sequence (Fig. 2).

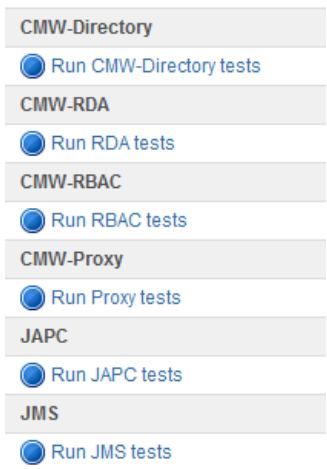


Figure 2: Bamboo's stages.

We use Bamboo to run the test suite of the Testbed every two hours. Since we use the JUnit framework to write our tests, full reports are also shown on Bamboo. Keeping the logs from each build is very handy because it happens that an error appears only once in a while, not at each run.

Using Bamboo to drive the Testbed has its limitations, the automatic deployment mechanism has to be scripted by hand (more about it in the next section), and setting up the environment such as JDK versions before running the tests is not obvious.

Automatic Deployment

As the Testbed is used for beta-testing, it is important that bugfixes can be deployed easily and quickly. We therefore invested in automatic deployment directly from the sources. Whenever a new component is ready, it is automatically built and deployed into the Testbed.

We use SVN branches to identify the source code that should be deployed into the Testbed. If Bamboo detects a commit to a specific branch, it automatically builds that branch and all the dependent projects as well. The resulting build artifacts are stored in a special binary repository, which eventually contains a set of components that have been built together. We added a post build mechanism to Bamboo, based on shell scripts, which deploys the executable artifacts into the Testbed. We use Apache Maven [4] for building, which works out-of-the box for Java. For C/C++ products, we have developed a Maven-compatible build system based on the Maven NAR plugin [5] that provides similar functionality as available in Java.

THE CASE OF CMW PROXY

This section explains in more details one particular component and the corresponding tests executed by the Testbed. The CMW proxy is a separate process used to shield devices on a front-end computer (FEC) from too many client requests. When several clients subscribe to a device through the proxy, it will manage all the subscriptions, but only do one subscription to the device. At first thought, the proxy seems to be a pretty simple piece of software, but in reality its functionality is rather tricky to implement, because it needs to be transparent to the clients and has many constraints due to RBAC security or FECs' implementation.

The middleware team, which is responsible for the proxy, updated a few of their C++ products in the release candidate repository, including the RDA communication library. They needed to deploy a new version of the proxy, which depends on RDA. Using the automatic deployment mechanism, the new proxy was updated and the tests worked fine during the first runs. But the next day, the tests were failing, because some proxied devices did not properly respond to requests. It turned out that the devices all worked fine, but the proxy was in a faulty state. We finally saw that this problem was caused by a newly introduced bug in RDA which did not properly close the connections. The Testbed had to run for 5 hours in order to reproduce this bug.

BENEFITS

The Testbed has been running for one year now and has already shown the following benefits.

More Confidence

Developers using the Testbed feel more confident in their product. The Testbed is an important complement to the unit tests, not a replacement. By running a series of tests after new software version, the Testbed ensures that a change does not break the core functionality of the controls system. The Testbed already caught several bugs and revealed few inconsistencies. The above example of

the CMW proxy product took several days to fix this bug and therefore saved some hours of beam downtime.

The Testbed is vital in our environment where many developers belonging to several teams contribute to the controls system. If one team provides a new version of their component, everyone can see immediately and well before operational deployment whether the controls system still works.

Better Understanding

The accelerator controls system is complex and hard to understand as a whole entity, hence the Testbed helps in that direction by simulating requests done operationally from the CCC. One can think of the RBAC implementation in combination with the proxy, in specific cases the proxy is overriding the client's permission. First we thought that it was a bug, but in fact it was done on purpose to force users to use the proxy.

Saving Money

The Testbed allows to validate the controls system core before it is deployed in the real accelerator complex. The overall cost of the Testbed (hardware and man power) is small compared to the cost of LHC downtime. Without having done a scientific analysis, we estimate that cost of the downtime avoided by the Testbed outgrows the cost of the Testbed itself.

Small Laboratory

The Testbed is a down-scaled replication of the accelerator controls system, which can serve as an experimental laboratory for many purposes. It was already used for early validation of new systems in an early stage of development. For example the new logging system which involves front-ends, middleware, the proxy and the JMS broker.

Another example is the new build and release tool we are working on, based on Apache Maven. It was first used in the Testbed, before even giving it to any of our Java and C/C++ developers.

FORESEEN IMPROVEMENTS

The most important improvement is to write more and better tests. Our developers should write functional tests in the same natural manner as they already write unit tests. We also have to extend test coverage to validate not only main functionality but also more advanced and less frequently used tests. It still happens (and we cannot avoid) that some bugs are discovered during operations. In these cases we need to enforce that a test is written to expose this bug.

As a second priority, we intend to extend the scope of the Testbed in several ways.

We will add new systems to be validated. At the moment, only the lower layers of the controls system are deployed into the Testbed. We plan to add higher-level core components, such as the Software Interlock System (SIS) [6], the LSA/InCA [7] system and the high-level settings management and controls system for our accelerators.

We will provide several different Testbed configurations with different versions of hardware, operating systems, Java virtual machines, and controls components. Functionality is needed to automatically re-configure the Testbed and re-deploy the controls system on it.

Finally, we might open up the Testbed to other types of tests than mere functional tests, and include performance and scalability tests.

REFERENCES

[1] N. Stapley et al., an integration testing facility for the CERN accelerator controls system, Proceedings of ICALEPCS'09 , Kobe, Japan.
[2] K. Sigerud et al., The Software Improvement Process – Tools and Rules to Encourage Quality, Proceedings of ICALEPCS'11, Grenoble, France.
[3] Atlassian Bamboo, http://www.atlassian.com/software/bamboo
[4] Apache Maven, http://maven.apache.org
[5] J. Nguyen Xuan et al., A C/C++ build system based on maven for the LHC controls system, Proceedings of ICALEPCS'11, Grenoble, France.
[6] J. Wozniak et al., Software Interlock System, Proceedings of ICALEPCS'07, Knoxville, USA.
[7] S. Deghaye et al., Cern Proton Synchrotron Complex High-Level Controls Renovation, Proceedings of ICALEPCS'09, Kobe, Japan.

AUTOMATED COVERAGE TESTER FOR THE ORACLE ARCHIVER OF WINCC OA

A.Voitier, P. Golonka, M. Gonzalez-Berges, CERN, Geneva, Switzerland

Abstract

A large number of control systems at CERN are built with the commercial SCADA tool WinCC OA (formerly PVSS) [1]. They cover projects in the experiments, accelerators and infrastructure. An important component is the Oracle archiver used for long term storage of process data (events) and alarms. The archived data provide feedback to the operators and experts about how the system was behaving at particular moment in the past. In addition a subset of these data is used for offline physics analysis (conditions data). Large volumes of data are produced by the different facilities at CERN (several Terabytes per year). The consistency of the archived data has to be ensured from writing to reading as well as throughout updates of the control systems. The complexity of the archiving subsystem comes from the multiplicity of data types, required performance and other factors such as operating system, environment variables or versions of the different software components. Therefore an automatic tester has been implemented to systematically execute test scenarios under different conditions. The tests are based on scripts which are automatically generated from templates, therefore they can cover a wide range of software contexts. The tester has been fully written in the same software environment as the targeted SCADA system. The current implementation is able to handle over 300 test cases, both for events and alarms. It has enabled to report issues to the provider of WinCC OA. The template mechanism allows sufficient flexibility to adapt the suite of tests to future needs. The developed tools are generic enough to be used to test other parts of the control systems.

INTRODUCTION

In a control system, archiving is responsible for recording the process data and alarms, and then making it available to understand the state a system was in at a certain point in time. The accuracy of the stored data in both value and time is a key aspect for the quality of service offered by an archiver. Operator tools such as trends or reports showing historical values rely on an efficient and accurate archiving system. Misbehaviours of these tools during a data taking period are not acceptable as they could lead to lowered operation capabilities, data loss or even a stop of the concerned system. In addition, upgrades of software components may only be applied at certain periods of time.

A particular requirement of the control systems for the experiments at CERN is to make use of subsets of data stored by the archiver as so called conditions data. The availability and consistency of such data, being for instance the monitored voltages of a high voltage power supply, are then essential and necessary for the off-line physics data reconstruction and analysis. In this example, archived data will act as an input parameter to calculate the actual gain factor of a photomultiplier, which may be linked directly to the efficiency or calibration factors of the detector.

To minimize the possibility of the down-time caused by a control system built with WinCC OA, the different components need to be validated by tests performed in laboratory conditions. This validation is handled by the CERN Industrial Controls group. The group provides a centralized service and support for common controls tools, and performs the initial validation of every new version of the SCADA software components. This include testing their proper operation in new environments (new versions of operating systems, related software, etc.).

Being the least mature and one of the most complex components of WinCC OA, the Oracle archiver requires particular attention. Not only do its scalability and throughput need to be verified (as previously reported in [2]), but also a systematic verification of consistency of archiving data across all possible data types and/or software context is required. The tool presented in this paper is aimed at this kind of coverage testing.

In the paper we will describe the Oracle Archiver, the problems associated with its testing and the approach we have chosen for the architecture of the tool. We will follow by describing the architecture of individual tests, and finally present conclusions.

The tools described in this paper are aimed at testing the behaviour of the Oracle archiver and to check on the integrity of data of various types, in multiple software environments.

To cover the large number of combinations of setups/environments that require testing, a program was devised to automate the – often tedious and complicated – process of setting up the archiving test setup in laboratory conditions. It became possible to repeatedly trigger the tests in the various environments and observe the impact of single environment changes on the result. This part of the project became an independent tool (known as Bootstrapper) that found many other applications.

The testing framework had to be generic and adaptive to cover a wide range of tests. The number of similar test scenarios called for an abstraction: the scenarios can be grouped in families of tests that differ slightly from one another. A powerful mechanism combines templates with test specific configurations to produce the test case

scripts. By using 8 templates and some minimal configuration, the current 300 scripts were produced.

Eventually, each issue discovered by the tester would have to be investigated, isolated and reported to the producer of WinCC OA. We came up with an architecture where every test case is self-contained and could eventually be executed with no additional instrumentation, and hence easy to reproduce in a clean laboratory setup.

Software Contexts

The tester is supposed to be repeatedly executed in varying setups (software contexts). It shall be possible to verify the proper functioning of the RDB Archiver in the setups that have different versions and patch-levels of WinCC OA software, Operating System (versions of Linux and Windows supported at CERN), Oracle software, or any other software component that may affect the functioning of RDB Archiver in the future. Other factors such as environment variable settings (e.g. the formatting of time representation, related to the local language setting), security settings, Oracle-specific settings or the Archiver's own settings affect the behaviour of the RDB Archiver and hence extend the spectrum of software contexts in which the tester need to be executed.

Context and Behaviour of the Archiver

The process of storing data is initiated asynchronously by the change of a value (that was declared as having archiving activated). The archiver employs a sophisticated buffering mechanism to optimize the performance with respect to the database, and groups the data before flushing them. As a result, the archived data is available in the database only after a certain "dead time".

The queries for historical data in WinCC OA are performed from scripts, by means of three functions: *dpQuery()*, a versatile SQL-like interface retrieving data, *dpGetPeriod()*, a more restricted interface, and *alertGetPeriod()*, oriented to alarms. The data for historical trend plots come internally through the mechanism used by the *dpGetPeriod()* function. Hence, for the purpose of testing, the functions cover also the trending functionality.

The archiver allows archiving values for the following WinCC OA basic data types: *char, uint, int, float, bool, bit32* (bit field), *string* and *time*. It also allows archiving array types made of the above basic types (*dyn_int, dyn_string*, etc.).

In addition to the value and time-stamp, the archiver also stores meta-data, such as the status bit fields, defining for instance the quality of the data. Such information can be used later in trend plots to visually mark invalid data.

An outstanding feature of WinCC OA is its very complex and flexible model for alarms. The archiver allows for the storing and querying of complete information about the evolution of alarms in the system. The alarm information is independent from the values, and it changes dynamically as the alarms come, go, are acknowledged and commented. The whole life-cycle of an alarm is traced by the archiver, and this domain also needs to be tested.

The Oracle archiver of WinCC OA has other functionality such as filtering, smoothing (reducing the data), correcting or compressing of archived values. These features have a direct influence on the archiving process. The archiver also allows for database specific maintenance operations such as grouping values in different tables, or taking a range of unused data off-line. These are not covered by the tester at present.

APPROACH AND ARCHITECTURE

The tester was developed iteratively following a progressive build-up of knowledge of the archiver's behaviour. It allowed us to focus the tests on the most critical points. A typical testing scenario was identified as a sequence of actions: trigger the storage of a value (or alert), then retrieve it from the archive, and finally compare the retrieved data with the expected (original), considering values, timestamps and meta-data (status bits).

As already mentioned in the previous chapter the individual test cases are embedded into individual test scripts coded in WinCC OA's own "Control" scripting language (CTRL). Each test script executes the above mentioned sequence of actions and reports the results.

To cover all possible combinations of standard test scenarios for all the WinCC OA data types, the scripts are generated from templates, based on configuration data.

For the automated tester executing numerous test cases in an already pre-configured environment we therefore established the following standard procedure:

1. Retrieve the configuration of the test case.
2. Generate the test script.
3. Execute the script.
4. Visually report and save the result of the test.

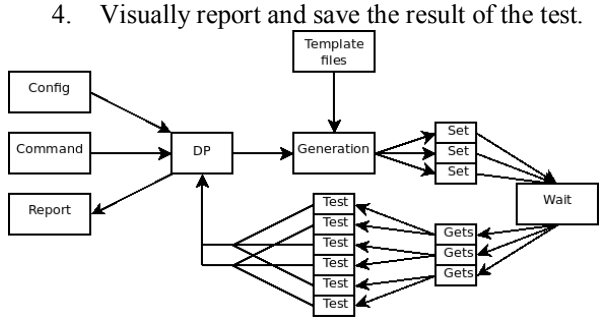


Figure 1: The tester design. (DP=Datapoint).

For all the operations of configuration, command and reporting (shown in Figure 1), the tester relies on WinCC

OA datapoints. In addition to being used to link to hardware and process variables, the WinCC OA datapoints represent variables with data-persistency and hence provide for easy inter-processes message passing.

Script Generation

The test scripts are generated whenever needed based on template scripts and adapted to each case through pre-registered configurations. Template files contain the basic structure needed to implement a test of a specific kind. Separate templates are required to address the differences between families of tests are instrumented for the particular data type (basic types, arrays, status bit or alerts) and the retrieval function they use (*dpQuery()* or *dpGetPeriod()*). The templates contain place-holders used during the generation process where they are adapted to the configuration of a test case. A configuration specifies the template file used, the name of the test, its subfamily (e.g. the retrieval function used here for display purposes), and a list of place-holder tags used within the template file. Typically, the place-holder tags allow specification of the data type of the test (*int, char, dyn_float*, etc.), the attribute of the datapoint the test targets (offline value, online value or original value), and the value used for this test.

Parallel Execution System

With a growing number of test cases the need for parallel execution became apparent. WinCC OA offers threading functionality inside the CTRL scripting engine. Due to current limitations these could not be used, therefore, a custom parallel-execution and job-queuing system free of these limitations was developed. The "script pooler" component employs a "pool" of individual CTRL language interpreter processes (managers) to schedule the execution of the subsequent test-case scripts. Each CTRL manager runs in a separate operating system process and is capable of communicating through datapoints, which allowed implementation of the parallel job queuing system. A component provides a panel to deploy and configure a list of CTRL managers for the pool. When deploying managers it is possible to configure the number of managers and pre-allocate them to individual test cases.

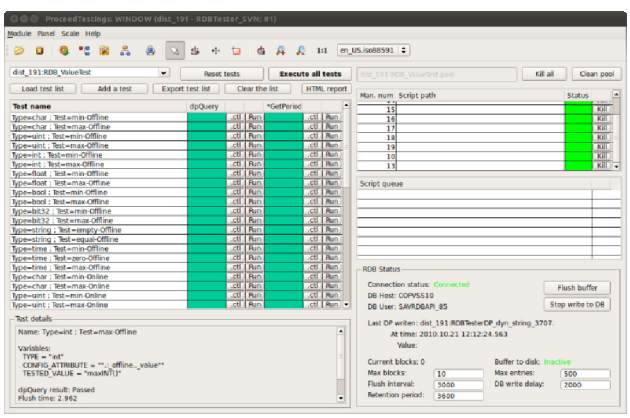


Figure 2: The overview panel of the tester.

Reporting

Once a test case has been executed, its script stores the result in a datapoint. It is then accessible later on, which allows running the tester overnight and checking the results later. The information that is stored includes: if the test passed or not, if it raised a warning, the database-flush time in case of success or the error reason otherwise. To display the results, an overview panel (see Figure 2) shows all of the relevant information for the progress of a test suite (all the tests of the same kind are executed in sequence). The panel also shows the script pooler (queue status, managers on duty, commands) as well as summary of the archiver status (latest datapoint written, buffer usage, connectivity, etc.).

TEST DESCRIPTIONS

Each test of the archiver is essentially made of a *Set* operation followed by a corresponding *Get* operation. The result of the *Get* being compared to what has been *Set* previously. Between the *Set* and the *Get* steps, however, the test has to wait for the data to go through the archiver's internal buffers and to get flushed to the database. Thus, we refer to this recurrent procedure as *Set - Wait flush - Get - Test*. Before being applied, this procedure needs to create a randomly named datapoint and configure it to be archived. The *Set* step is performed such that the timestamp on the archived data change is known.

The *Wait* step needs to be synchronized between the tests to reconcile with the buffering behaviour of the archiver. To compensate for a missing functionality within the archiver, a mechanism was implemented to confirm and notify when the archived data reached the database, or that a timeout expires. The *Get* step will then retrieve the previously archived data, using the WinCC OA data-query functions, and extracts the details necessary for the *Test* step. This last step compares the timestamps, the values, the types and the status bits.

For the eight scalar data types, each type is tested in a separate test case. Each test case performs a number of sub-tests, e.g. for successive data point elements (the offline, online and original values), and data-retrieval functions (*dpQuery()* and *dpGetPeriod()*). Finally, several value variations are introduced (e.g. maximum/minimum value for a particular type, positive/negative values). It would be excessively large and not meaningful to test every single possible value for each data type. Therefore only the *min* and *max* functions for numerical types, and an empty and a random value for the string type are tested.

For arrays the tested variations are similar to the scalar values. The special case of an empty array is handled. The size of an array, the type of its elements and the order of the elements are checked.

For the test-case of status bits the original status set is compared to the one retrieved; the actual value and the

data type variation are not the subject of the test. Different realistic status values as well as different archiving contexts such as smoothing (data reduction) are set and tested.

For alerts the tests cover the retrieval functions and datapoint attributes, and also the alert configuration itself (number of alarm ranges crossed, type of alarm (value range, discrete value, status bits)). Several scenarios were implemented for alert evolution, with permutations of the *Came, Went* and *Acknowledge* events in the alert life-cycle. Although alerts can display a hysteresis effect because of this sequence of events, the creation of a random test datapoint at the beginning of each test ensures that there is no perturbation between concurrently running alert tests.

RESULTS

An issue with flushing of the archiver's buffer was identified and isolated as soon as we began production testing. Thanks to the tester, the issue which was previously only seen sporadically in production systems could be studied, reproduced and reported to the WinCC OA vendor, and resulted in a fix being made available for CERN users.

The tester identified seven problems with data consistency for scalar data types, namely for special cases such as empty strings or ASCII-0 characters, mis-handling of timezone-setup in the operating system, or bit-field incoherence. Thanks to the requirement of the test-cases being self-contained, it was easy to demonstrate the problems to the vendor, and all of the issues were quickly resolved. Since then, the tester has been reporting the status of all tests for scalar values as "green" (passed).

The tests for array-types detected a further nine issues, which – after being studied – will also be reported to the WinCC OA vendor. Again, problems come up in the handling of non-standard cases such as empty arrays or arrays containing empty strings.

The tests for the status bits did not detect any problems. Nevertheless, the tests reassured us that the problems encountered in previous versions of WinCC OA have not re-appeared. The tests for alerts have not detected any inconsistency so far.

The tester is now routinely used to check all new releases (patches) of WinCC OA that affect the archiver subsystem. Reliable and easy-to-deploy tests significantly reduce the workload on the WinCC OA support team at CERN.

DISCUSSION

The automated coverage testing tool allowed to detect, reproduce, isolate and report a number of issues with the WinCC OA Oracle Archiver, and have them resolved in time for the 2009 run period of the LHC. Similarly, it allowed sorting out an issue with the buffer-flushing mechanism, which affected to the operation of some control systems already in production.

Test cases for array-types have been much more challenging. Firstly, they are harder to debug considering their size-related and ordered nature. Problematic behaviour sometimes appeared only with a specific order of elements. The archiving of array-typed values is nevertheless a feature rarely-used at CERN. Therefore testing and correction of its mechanism was not top priority. However, the results of failed test cases will be reported to the WinCC OA vendor to be corrected in a later version of their product.

For alerts, we decided to not make any further development of the tester until we establish a more efficient testing structure, as the alert life-cycle scenario is too complex for the current approach.

CONCLUSION

In this paper we presented the automated tester for Oracle Archiver of WinCC OA. The tester proved its usefulness in detecting and isolating problems with the archiver, by systematically testing the archiving functionality for all supported WinCC OA data types. Together with the previously discussed scalability-tests [2], the results provided by the automated coverage tester build up our clients' confidence in the Oracle Archiver.

The issues that were easily discovered by the tester confirmed that the chosen approach was the right one. Furthermore, the technical decision of implementing our own parallel-execution system brought additional advantages: a solution offering native parallelisation of CTRL scripts in WinCC OA, and a way to easily isolate the test cases into self-contained scripts, simplifies reporting the issues. Finally, the "Bootstrapper" used to prepare and start the tester evolved into an independent project and finds application beyond its original scope. We believe that the components of the tester may also be used to leverage the testing of other components of WinCC OA, such as the alert system, or the frameworks used to construct the controls applications at CERN.

REFERENCES

[1] ETM, "ETM professional control GmbH - A Siemens Company - WinCC Open Architecture - SCADA System."; http://www.etm.at.

[2] The High Performance Database Archiver for the LHC Experiment. ICALEPCS07, Knoxville, Tennessee, USA.

AUTOMATED TESTING OF OPC SERVERS

B. Farnham, CERN, Geneva, Switzerland

Abstract

CERN relies on OPC Server implementations from 3rd party device vendors to provide a software interface to their respective hardware. Each time a vendor releases a new OPC Server version it is regression tested internally to verify that existing functionality has not been inadvertently broken during the process of adding new features. In addition bugs and problems must be communicated to the vendors in a reliable and portable way. This presentation covers the automated test approach used at CERN to cover both cases: Scripts are written in a domain specific language specifically created for describing OPC tests and executed by a custom software engine driving the OPC Server implementation.

INTRODUCTION

The architecture of many of CERN's Detector Control Systems (DCS) is, to a large degree, stable. Users of the DCSs are currently able to sufficiently control and monitor their systems to successfully create and gather physics data. Nothing is perfect however. Controllers and administrators have working systems, but there is scope for improvement: fixing bugs, adding features, improving response times to commands, reducing feedback latency etc.

The OPC Test Script Runner is a tool, developed at CERN, for writing, running and recording the results of scripted tests which engage DCS components at the level of OPC [1] (a standard providing a uniform means of controlling and monitoring devices). The tool aims to address two concerns regarding the stable DCS environment described above:

- Changes carry an inherent risk of regression: Changing X might accidentally break Y (and Z). A DCS user may request a new feature to a device, exposed via its OPC Server. The user wants assurance that, after making changes, the critical features on which the DCS relies function at least as well as they did before. Additionally the user wants assurance that the new feature functions as per specification and will continue to do so over time as subsequent modifications are made.
- It can be hard to accurately convey a bug to a vendor, and hard for a vendor to recreate a bug for diagnosis. Consider a subtle bug in an OPC Server, occurring only under some long and complicated sequence of client operations: Communicating the detail of this bug to the vendor involves the user translating this sequence into a textual description, transmitting it to the vendor who de-translates that description back into OPC client actions to replay to an OPC Server in an attempt to reconstruct the original bug. Written language can be an awkward medium for unambiguously recording a series of complex actions. Misinterpretation of one action in the sequence may result in the vendor being unable to recreate the problem for diagnosis.

OPC test scripts describe functionality and perform runtime verification. Test scripts, designed to capture behaviour of critical features, are run against new releases to check whether the modifications damaged existing functionality. Furthermore, as new features are released, their functionality is described and verified through test scripts in order to test the feature at point of release. These test scripts are added to the catalogue of regression tests to verify that functionality subsequent releases.

A similar approach applies to bug reporting - in effect a test script is written which fails, the failure highlighting the ill effects of the bug. The script is passed to the manufacturer who runs it to observe the bug first hand. The bug is fixed once the script passes.

THE TEST SCRIPT RUNNER STRUCTURE

The OPC Script Runner contains an OPC client. At their most basic level, OPC test scripts describe sequences of OPC interactions which the client carries out with the server at runtime. A test script which simply specifies a sequence of client interactions is not sufficient however, in order to ascertain whether the interactions had the desired effect, i.e. whether the test script passes or fails. There is an additional tie between the test script and script runner, namely assertions, a notion familiar to unit testing practitioners. Assertions in the script instruct the assertion handling module of the script runner to set watchpoints with specific criteria for success, failure to meet these criteria results in the assertion failing, and thus the test script failing. The test script runner records and displays assertions results. Assertions are described later in more detail.

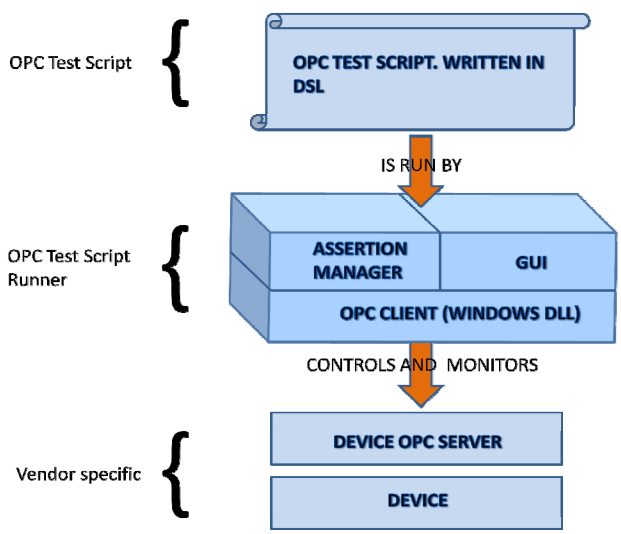


Figure 1: OPC Test Script Runner Components.

The OPC Client part of the test script runner is a MSWindows dll, written in C++, the upper part of the test script runner, handling script execution, assertion management and the graphical user interface is written in Groovy, a dynamic language for the Java Virtual machine. Interactions between the higher level processing on the JVM and the lower level OPC client are handled by open source Java Native Access (JNA).

A dynamic language such as Groovy was an important choice for the upper portion of the test script runner in order to aid defining the Domain Specific Language [2] (DSL) in which the test scripts are written. DSLs are 'small languages', designed for a particular field and consisting of nouns and verbs specific to that field. The field in this case is OPC testing, OPC nouns from the DSL include familiar OPC terms such as 'group' and 'item' and verbs pertaining to OPC nouns like an item's 'asyncWrite' or a group's 'destroy' plus unit test style verbs such as 'assertTrue' or 'assertNotEqual'. The DSL for the test scripts has been designed to promote readability, the idea is that 3rd party users such as an OPC Server vendor should be able to read a testscript and understand it in terms of the OPC interactions.

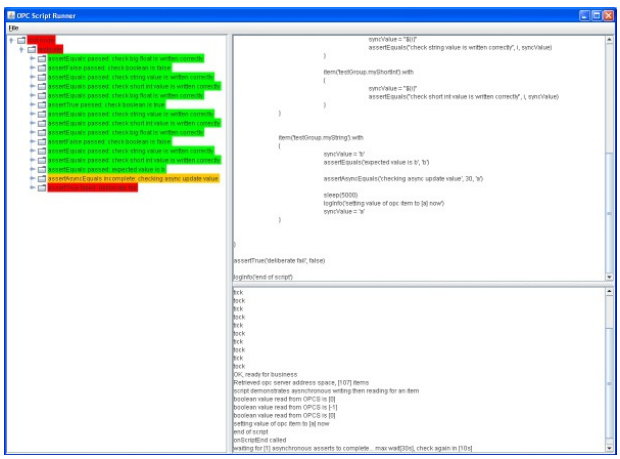


Figure 2: OPC Test Script Runner GUI layout.

The left panel displays a tree of the script's assertions colour coded on their pass/pending/fail states, the tree is dynamically built at runtime as the assertions are made and the assertion criteria met or otherwise. The top right panel displays the script text (read only) and the bottom right panel displays a log window which updates with system messages and user defined log messages written in to the script. From the menu users can open and run scripts and export script assertion results in Junit style XML format.

ASSERTIONS

Assertions are the mechanism for defining pass/fail criteria in test scripts. Two types of assertions are available:

Synchronous Assertions

These are the simplest case. A script can make assertions about an OPC item's immediate value - that it is equal to x, or not equal to y for example. Synchronous assertions pass or fail immediately. Script excerpts detailing synchronous assertions include:

- *item.assertEquals("Verifying the value of an integer item", '50')*
- *item.assertTrue("Verifying the value of a boolean item")*
- *item.assertNotEquals("Verifying the value of a string item", 'ERROR')*

Asynchronous Assertions

OPC interactions can be asynchronous in nature. For example OPC has the concept of groups: A client requests that a server builds a group and adds items to it and that the server informs the client of changes to group items at a rate not greater than some client specified frequency. From the perspective of the test script, commands can be delivered through the client but the effects only seen some time later. This use-case is handled by the asynchronous class of assertions. These assertions provide a required criteria for success (as with their sycnhronous siblings) however they also specify a time limit within which the criteria must be met. Asynchronous assertion excerpts from scripts include:

- *item.assertAsyncEquals("Waiting a maximum of 2.5 seconds for channel status to be stable:on", 2500, 1)*

Asynchronous assertions have 3 states: Pass, fail and pending (while the criteria is not met but the time limit not yet reached). This asynchronous class of assertions allow for performance test scripts. For example an asynchronous assert could be scripted immediately after multiple commands have been sent to verify whether a system remains responsive (to within the timeframe specified by the assertion) following a command flood.

A SIMPLE EXAMPLE SCRIPT AND ITS ANATOMY

The following is an OPC test script used for verifying the functionality and response time for turning channels on for a CAEN industrial power supply. Note some parenthesis and user messages have been omitted for brevity. As previously stated, the DSL in which test scripts are written is intended to be readable by 3rd parties such that they can understand what the test does and the assertions the tests make. The DSL includes a regular expression like syntax (*) to denote collections of OPC items with matching addresses.

```
init('', 'CAEN.HVOPCServer')

group('setup.software.and.hardware.chain').with
   item('**.ConnStatus').assertEquals('Ok')
   item('**.OPCServerEventMode').assertFalse()

group('set.initial.device.state').with
  items('**.Board*.Chan*.Pw').each
```

```
        it.syncValue = 'false'
items(**.Board*.Chan*.Status').each
        it.assertEquals('0')

group('main.body').with
   // set up asynchronous assertions
   items(**.Board*.Chan*.Status').each
        it.assertAsyncEquals(10000, '1')
   // turn the channels on
   items(**.Board*.Chan*.Pw').each
        it.syncValue = 'true'

// pausing for 11s to allow asynchronous criteria
sleep(11000)

group('wrap.up').with
   items(**.Board*.Chan*.Pw').each
        it.syncValue='false'
logInfo('end of script')
```

The test script runner dictates no special structure for the test scripts it executes, however experience has shown the structure followed above to be an effective skeleton:

1. Verify the software and hardware chain: Assert that the OPC Server and underlying device are present and the basic state is suitable. For example is there an OPC item which can be used to assert that a device is connected to the OPC Server and is powered on.
2. Set and verify the pre-test device state: Send commands to the underlying device to set the initial state and assert that the required initial state has been achieved.
3. The main body: Commands are issued and assertions made about the effects of those commands.
4. Wrap up: Send commands to set any post test state and assert that the post test state is achieved

USE CASES

OPC test scripts are simplest when a repeatable sequence of events and their corresponding effects can be defined. For regression test type scripts this is almost always the case: A known sequence of client/server interactions is expected to have a known set of required outcomes. So long as these outcomes are visible to the script runner via OPC then they can be verified in the form of assertions. A slightly more complicated type of test involves a known end effect (observable via OPC) but with an undefined set of events leading up to it - often this is the situation regarding subtle bugs: For example after many hours of continuous operation an OPC item suddenly ceases to update. Again, so long as the effect is observable via OPC the behaviour ought to be able to be captured using an OPC test script. Test scripts can request random numbers to introduce a stochastic element into interactions.

Some examples cases where the OPC Test Script Runner has already been employed:

1. A device vendor must switch their OPC Server implementation from an outdated OPC toolkit library supplier to another (in order to provide Windows 7 support). This change is not insignificant. A catalogue of test scripts has been written against the previous OPC Server version on XP. These scripts are being run against the new OPC Server version on windows 7 to check for bugs and deterioration in functionality.
2. The OPC test script runner has been used to successfully narrow down operations causing a memory leak in an OPC Server from an industrial power supply vendor. Memory usage is not available via OPC so this was monitored using an external tool: Windows XP Perfmon. Multiple scripts were written, each script focusing on a different type of client/server interaction (different types of reads and writes for example), and each script run for some set period whilst monitoring the memory usage. Certain scripts appear to cause the OPC Server memory usage to grow more quickly than others, providing the vendor with empirical information as to the problematic client/server interactions. The scripts (plus runner) have been passed to the vendor.

FURTHER WORK

Currently scripts are a single continuous sequence. The ability to define methods (parameterized repeat blocks of script) would allow for more concise and intuitively readable scripts.

The current implementation of the OPC Test Script Runner supports only the OPC Classic specification released in 1996 and based on deprecated Microsoft COM/DCOM technology. The OPC Foundation released a next generation OPC specification called OPC-UA [3] which looks set to gradually replace OPC Classic over time. The OPC Test Script Runner client will be updated to be OPC-UA compliant and the test script DSL extended to include OPC-UA operations.

The test scripts runs without human intervention, however, each script must be loaded and started by hand. It would be more efficient to instruct the script runner to run a catalogue of tests sequentially. For example run all scripts in a given directory.

CONCLUSION

Automated testing is an established practice in software development, providing increased assurance against the regression of existing functionality as current OPC based DCS technology evolves. Furthermore, the ability to provide vendors with runnable instances of bugs and problems can provide significant efficiency gains over writing traditional bug reports. The OPC test script runner provides a means to bring both of these benefits to bear on the field of OPC components.

REFERENCES

[1] The OPC-DA Specification.
www.opcfoundation.org

[2] M. Fowler and R. Parsons “Domain-Specific Languages”. Addison Wesley 2011.

[3] The OPC-UA Specification
www.opc-foundation.org/ua

BACKWARD COMPATIBILITY AS A KEY MEASURE FOR SMOOTH UPGRADES TO THE LHC CONTROL SYSTEM

V. Baggiolini, D. Csikos, P. Tarasenko, Z. Zaharieva, M. Arruat, R. Gorbosonov,
CERN, Geneva, Switzerland

Abstract

It is a big challenge to smoothly upgrade the control system of a large operational accelerator such as the LHC without causing unnecessary downtime. We have identified backward compatibility as a key measure to achieve this, because a backward compatible component can be easily upgraded. This document describes the work the CERN Accelerator Controls group does to provide methods and tools supporting backward compatibility.

BACKGROUND

Now that the LHC is operational, we from the controls group get requests from the operations team, which require a high degree of versatility. On one hand, the stability of the control system is necessary to ensure smooth operations while on the other hand, a certain amount of flexibility should be available in order to develop and deploy bug fixes and new functionality. To cope with these requirements, we are making a continuous investment in the quality assurance of our software, to improve the development process [1] and to provide new tools [2]. The work described in this document belongs into this context.

The CERN accelerator control system is highly complex, modular and distributed. The software part is structured as a three-tier system. GUIs at the top layer are written in Java and run on Linux consoles in the control room; the business layer in the middle is also written in Java but runs on powerful Linux servers in the computer center; the equipment control software at the lowest layer is written in C/C++ and runs on front-end computers with real-time-enhanced Linux and LynxOS. The Java part of the control system is composed of roughly 1000 Jar files, which are combined to around 400 different GUIs and 150 different server programs, which are deployed as over 600 processes on 400 machines. The C/C++ part on the front-end computers is represented by around 550 different device types (FESA [3] and legacy GM classes), deployed as over 70'000 device instances on 800 front-end computers. The development of all these components is done by more than 130 developers in 50 different teams. The development and collaboration is organized in a pragmatic and informal manner, with very low administrative overhead. Developers essentially collaborate along the dependencies of their software. If two components depend on each other, the respective developers will coordinate their work as needed. Even though there is no strong centralized organization to coordinate all upgrades done by the different teams, this form of collaboration is very efficient and agile. However, there are certain shortcomings as developers are not always aware of all the other components that depend on theirs, and consequently they may fail to fully coordinate their work with everyone. Therefore, an upgrade may result in down time to the LHC.

THE DEVELOPER'S VIEW

Let us examine the problem from the perspective of an experienced developer, who needs to modify a widely used library, which possibly requires changes in the signature of an API method.

Let us examine the steps this developer will follow. As a first step, he* will decide if he may modify the API method or not. He will try to understand if some other component uses that method, i.e., he will examine number and origin of *incoming dependencies*. There are three possible cases: there are no incoming dependencies, just a few of them, or many. In the first and the last cases, the situation is clear. If there are no incoming dependencies, then he can freely modify the method, because he has no backward compatibility constraints. If there are many incoming dependencies from many client components, then he cannot change the method signature; he must modify his library in some backward compatible way.

If there are just a few incoming dependencies from one or two client components, the developer can chose between two approaches: He can either accept backward compatibility constraints or he can break backward compatibility. The two solutions have opposite advantages and disadvantages. In the first case, staying backward compatible makes development more difficult but deployment easier. Backward compatibility constraints (keeping the old method signature) typically lead to sub-optimal solutions with more code to maintain. Also, the developer has to validate that the change is really backward compatible, which might be difficult. Deployment is easier because a backward compatible component can be deployed anywhere without breaking any dependent clients. Deployment can be done selectively, starting with those systems that really need the new functionality, and deferring upgrades of the other systems until the upgrade has been validated. There is no need for wide coordination or big-bang changes.

In the second case, advantages and disadvantages are inverted: breaking backward compatibility makes development easier but deployment more difficult. The developer has no backward compatibility constraints, and can choose the best solution, which generally leads to cleaner results and less maintenance. However, careful

* This article uses the masculine pronoun 'he' for brevity, but intends 'he or she'

coordination is needed with other developers responsible for dependent components. They must change their source code to adapt to the new API and re-build and re-test their components. And then, new versions of all components must be deployed at the same time. There are two issues with that: firstly, all this can be difficult to organize, and secondly, the other developers will be unhappy if they need to adapt, rebuild, re-test and re-deploy their client components too often.

The above description admits that our developer has a clearly defined API and that his clients respect this API. Reality though might be different. The developer may not have clearly specified which classes and methods belong to the API, and the clients may disregard the official API and use additional (non-API) classes contained elsewhere in the software component. As a consequence, with an insufficiently specified and enforced API, our developer has to take the precautions above for each and every public method he might want to change.

TOOLS TO SUPPORT SMOOTH UPGRADES

The description above illustrates that upgrading a widely used component is a challenging development task that must be supported by good tools. We have identified four areas for which we want to provide tools: (1) dependency analysis to identify incoming dependencies, (2) backward compatibility validation to verify that API changes are really backward compatible, (3) versioning with rules to clearly inform the dependent clients if a modification is backward compatible, and (4) API consolidation to clearly specify classes and methods belonging to the API and to enforce their appropriate usage.

The following subsections discuss each of these areas, by first presenting existing approaches and tools and then motivating and explaining own developments we did.

Tools to Analyze Incoming Dependencies

Our developer wants to know about incoming dependencies right from inside his IDE. For example, he wants to right-click on a given method and execute a command “show incoming dependencies”. As a result, he expects to get a list of client libraries that use the selected method, with the possibility to navigate to the client source code from where the selected method is invoked.

Most IDEs provide very similar functionality, namely to show the call hierarchy of a method within a given project. However, this functionally only reveals incoming dependencies from within the source code and components (Jar files) present in the IDE. It does not show incoming dependencies from external Jar files. Also, it does not take into account previous versions of components that are deployed in operations, but only shows dependencies between the latest snapshot of the source files.

There exist stand-alone tools capable of analyzing dependencies between a large set of Jars, such as Tattletale [4] and JDepend [5]. They produce a lot of useful information, but they analyze dependencies only at the class level, not at the method or field level as we need.

Therefore, we decided to develop our own tool which is based on a client-server architecture. The server creates a list of all the roughly 1000 Jar files running in production and analyses the byte code of all classes they contain using the Apache Commons Byte Code Engineering Library BCEL [6]. It collects information about method and field-level accesses from one Jar file to another. The dependency information is stored in a database, and made accessible to the client over a remote RMI call. The client is an Eclipse plug-in with the right-click-on-method functionality described above. The analysis is carried out every 20 minutes and takes roughly a minute to run.

Tools for Assessing Backward Compatibility

Our developer also needs tools which help him ensure that modifications he made are really backward compatible. Ideally, backward compatibility is verified early on, during development, warning the developer as soon as he breaks backward compatibility. We first concentrated on tools to assess *binary* backward compatibility for Java. After all, the goal is to deploy a new version of a component (a Jar file) without even recompiling the client code. The Java Language Specification [7] contains clear rules to guarantee binary backward compatibility.

The “PDE API tools” [8] contained in the Eclipse IDE provide exactly this functionality. They assess backward compatibility of a project in the IDE by comparing it with a previous version of the same project. Once this functionality is activated and properly configured, the IDE gives immediate feedback and warnings about backward compatibility violations to the developer. Although this looks like a perfect fit to our needs, there are two issues to overcome. Firstly, PDE API tools (and Eclipse as a whole) use OSGi [9] to declare the public API packages. So far, we do not use OSGi, and introducing OSGi into a software development process is a big decision, which should not be driven only by the needs of one tool. Secondly, a considerable amount of manual configuration is required to properly configure the PDE API tools. We would have to automate this because we cannot expect our developers to do it manually. Therefore, we will need to weigh the benefit of using this tool against the overhead just described.

Of course, comparing the API signatures of different versions is only the first step for checking backward compatibility. It can be considered as a sort of early-warning system for the developer while he modifies his code. To validate backward compatibility further, we rely on other means, such as function tests and our Continuous Integration server. We also validate the core elements of the control system in the Controls Testbed [2], which carries out function and integration tests. In the future, we might also explore more formal approaches to augment the API specification, such as Design by Contract [10].

Versioning Schemes and Related Tools

Once our developer has finished the modifications to his component, he has to increase the version number. All the Java components in the accelerator control system are versioned using a scheme with three numbers separated by dots (*x.y.z*), which are called *major*, *minor* and *micro* (*major.minor.micro*). Semantic versioning [11] assigns a meaning to each of these numbers: if the developer did a (backward compatible) bugfix, he increases the *micro* number, if he added functionality but the overall change is still backward compatible, he increases the *minor* number, and if he breaks backward compatibility, he increases the *major* number. If the developer uses semantic versioning, the clients can simply infer the impact of the change from the change in version number. A typical client will automatically accept new versions of a library if only the minor or micro numbers have changed.

We want to start using semantic versioning, and back it up with tools that automatically calculate the new version based on the changes made to the code. We have identified two tools providing such functionality: PDE API tools and a semantic versioning plug-in for Maven [12].

Tools to Specifying and Enforce APIs

To fully define a Java API, our developer must be able to specify the packages that contain API classes. In addition, he may add further constraints, e.g. to indicate that clients are allowed to use a given public interface (invoke its methods) but not to extend it.

Once an API is specified, it must be enforced, e.g. clients must be prevented from using non-API classes. Standard Java does not provide sufficient mechanisms for this purpose. Therefore, PDE-API tools use OSGi to specify and enforce access to API packages, and Javadoc tags to specify further API constraints. For example, the @noimplement tag indicates that an interface can be used but should not be implemented in client code.

A completely different approach is based on static crosscutting functionality provided by AspectJ [13]. The 'declare warning' construct of AspectJ makes it possible to issue warnings for illegal method access, e.g. a client accessing non-API methods. These checks are executed at compile-time, and simply require the AspectJ compiler and weaver to be executed as part of the build process.

APPLYING THE SAME CONCEPTS TO C++ CODE AND FESA DEVICES

So far, this document has only discussed Java software. The other languages we use are C and C++. We started with Java because this is the area where we need to achieve smooth upgrades first, and because Java provides many mechanisms and tools for what we want to do. Once we have a clear idea about our needs and feedback from developers, we intend to provide similar tools for C/C++.

But already now, we try to use backward compatibility concepts for the low-level software that integrates the hardware devices into the accelerator control system. As for Java, we need to provide guidelines and tools enabling the device developers to achieve smooth upgrades.

All our devices follow the device/property model, which means that a device class has properties (e.g. a Magnet has a Current property, or a Motor has a Position property). The public API of a device class is represented by the properties it exposes to its clients.

Device developers modify the public API from time to time, to make bug-fixes or to introduce new functionality. As for the Java libraries, these modifications can be backward compatible or not. For instance, adding a new property to a device class is a backward compatible modification, whereas renaming an existing property breaks backward compatibility.

Currently the versioning of FESA devices is slightly different from versioning Java software. FESA device classes do have a version, but the device developer is expected to increase it only if his modifications break backward compatibility. He can keep the version unchanged if his modifications are backward compatible. Whenever the version is increased, client code accessing the devices must be re-configured or even re-released to use the new version of the FESA class. If the version is unchanged, the modified FESA device just replaces the old one operationally as soon as it is deployed.

With the current FESA tools, the developer is only *expected* to increase the version when he breaks backward compatibility, but not *forced* to do so. This has occasionally lead to problems in the past. Developers have made non-backward compatible modifications to their FESA device/property API without increasing the version number. After the deployment of these devices, some important client applications stopped working because they relied on the old API, and LHC operations were affected.

We are now improving the development process and tools for FESA development based on the same concepts as described above for Java software development. The following paragraphs shall explain this in more detail.

A FESA developer, who needs to make a modification to his device, carries out the same steps as the Java developer discussed earlier. He first analyzes the incoming dependencies to the device property he might want to change. He then decides whether to stay backward compatible or not. In case he remains backward compatible, he needs tools to validate backward compatibility, and so on.

Regarding the tools, we are developing means to assess incoming dependencies to the device properties of FESA classes. This work is largely based on the Controls Configuration Database (CCDB) [14], which contains the configuration data for all device classes, their properties and devices. It also contains configuration of the most important client applications that use the FESA devices, such as LSA and InCA [15]. Thus, the accelerators controls system is to a large extent data-driven and the

challenge is to ensure that a coherent set of configuration data is used throughout the controls system. To achieve that, we have established a strategy with several steps. First of all we need to clearly specify the visibility of the configuration data. Previously all data – data necessary for the operation of the accelerators and data used by equipment experts for low-level device control – was exposed to everyone through public APIs. Now the approach has changed and the data for the FESA classes is divided into a public API used for operations and a private API used only by equipment experts. Another important aspect is that the CCDB needs to have feedback of who uses what from the public API, in order to ensure that only backward compatible changes are made. This feedback allows us to find out which device properties are used in operations, i.e., determine the *incoming dependencies* for a given device property.

However, not all clients of FESA devices propagate feedback to the CCDB regarding the device properties they use. We have important systems that use configuration files or other means to specify the device and properties they need, e.g. the Software Interlock System [16] or the Post Mortem Analysis system [17]. We are currently examining methods to dynamically harvest information about device property usage for all Java client applications. For this purpose, we have instrumented the JAPC communication library [18], which is used for nearly all device access from Java. JAPC dynamically tracks the device properties that were accessed and periodically uploads this information to the CCDB.

Validation of backward compatibility during development is done by the newest FESA development tools, which compare the previous version of the public device/property API of a FESA class with the one under development. If the FESA tool detects a change that is not backward compatible, it forces the developer to increase the version number according to the semantic versioning rules. This should prevent the problems described earlier in this section.

Finally - unlike Java - FESA does not need additional tools to specify and enforce the public APIs, because this is already covered very well by the existing FESA tools.

CONCLUSIONS

We need to upgrade the LHC controls system without causing any unnecessary down time. Backward compatible changes are a means of achieving this. The same concepts of backward compatibility apply to Java software and to FESA devices. In both cases, making backward compatible changes is challenging, and developers need to be supported by good tools. We have identified some promising approaches and tools, and we know how we want to apply them. We are evaluating the different alternatives and have started implementing our own tools where third party solutions are missing or insufficient. This is a non-negligible investment, but we are convinced that it will pay off, by making our work easier and the LHC operations more reliable.

REFERENCES

[1] K. Sigerud et al., "The Software Improvement Process – Tools and Rules to Encourage Quality", Proceedings of ICALEPCS'11, Grenoble, France.
[2] N. Stapley et al., "An integration testing facility for the CERN accelerator controls system", Proceedings of ICALEPCS'09, Kobe, Japan.
[3] M. Arruat et al., "Front-End Software Architecture", Proceedings of ICALEPCS'07, Knoxville, Tennessee.
[4] Tattletale, http://www.jboss.org/tattletale
[5] JDepend, http://clarkware.com/software/JDepend.html
[6] Apache Commons Byte Code Engineering Library, http://commons.apache.org/bcel/
[7] J. Gosling et al., "The Java Language Specification, Third Edition", Addision Wesley 2005, http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf
[8] PDE API tools, the Eclipse Foundation, http://www.eclipse.org/pde/pde-api-tools/
[9] OSGi™, the Dynamic Module System for Java, http://www.osgi.org/
[10] Design by Contract, http://en.wikipedia.org/wiki/Design_by_contract
[11] Semantic versioning, http://semver.org
[12] Semantic versioning plugin for Maven https://github.com/jeluard/semantic-versioning
[13] R. Laddad, "AspectJ in Action", Manning 2010. http://www.manning.com/laddad/
[14] Z. Zaharieva et al., "Database Foundation for the Configuration Management of the CERN Accelerator Controls Systems", ICALEPCS 2011, Grenoble, France.
[15] S. Deghaye et al., "CERN Proton Synchrotron Complex High-Level Controls Renovation", Proceedings of ICALEPCS'09, Kobe, Japan.
[16] J. Wozniak et al., "Software Interlock System", Proceedings of ICALEPCS'07, Knoxville, USA.
[17] M. Zerlauth et al., "The LHC Postmortem Analysis Framework", Proceedings of ICALEPCS'09, Kobe, Japan.
[18] V. Baggiolini et al., "JAPC - the Java API for Parameter Control", ICALEPCS'05, Geneva, Switzerland.

SOFTWARE TOOLS FOR ELECTRICAL QUALITY ASSURANCE IN THE LHC

Mateusz Bednarek, CERN, Geneva, Switzerland,
Jaromir Ludwin, IFJ-PAN, Kraków, Poland

Abstract

There are over 1600 superconducting magnet circuits in the LHC machine. Many of them consist of a large number of components electrically connected in series. This enhances the sensitivity of the whole circuits to electrical faults of individual components. Furthermore, circuits are equipped with a large number of instrumentation wires, which are exposed to accidental damage or swapping. In order to ensure safe operation, an Electrical Quality Assurance (ELQA) campaign is needed after each thermal cycle. Due to the complexity of the circuits, as well as their distant geographical distribution (tunnel of 27km circumference divided in 8 sectors), suitable software and hardware platforms had to be developed. The software combines an Oracle database, LabView data acquisition applications and PHP-based web follow-up tools. This paper describes the software used for the ELQA of the LHC.

INTRODUCTION

A great diversity of circuits and configurations met in the LHC machine forces the use of several software technologies within one application.

This quality assurance system was developed in parallel with the machine assembly. It was expected to verify the correctness of many parameters that have never been measured before and thus be able to quickly adapt to the new circumstances. Therefore, the set of measurements and the acceptance criteria were expected to change as the installation and development works were progressing.

A flexible software platform had to flawlessly drive the dedicated hardware [1], stay in accordance with the machine layout and safely store the measured values, while allowing a central management of the full software structure.

TEST FAMILIES

The applications have been divided into three families: TP4 (Test Procedure 4) for superconducting circuits powered via the DFBs (Distribution Feed Boxes), DOC (Dipole Orbit Corrector) for locally powered superconducting orbit correctors and MIC (Magnet Instrumentation Check) for local magnet voltage pickups and quench heaters. Hardware setup for each type of tests has been described in [1].

DATABASE

A database system constitutes a core of each application. It is used in order to read application settings and store measured values. As there is no access to the network at the measurement positions each system is working on a local database (MS Access). After coming back to the surface infrastructure it synchronises its local database with the centrally hosted Oracle storage: the measured values are copied in the Oracle structure; all the settings used for the tests in the tunnel as well as the most recent layout of the machine are copied from the Oracle to the local database.

Communication between LabVIEW applications and databases is done using ODBC technology.

Some tables located in the central database have no correspondence in local MS Access files. Those tables are used by web applications. Central database includes also a number of synonyms which are pointing to views in LHC Layout database. Such synonyms are used as reference data sources for all our applications. Currently central database consists of over 80 tables.

DATA ACQUISITION

The hardware part of the measurement system as well as the user interface for the test benches are driven by a set of LabVIEW applications. Families of aforementioned applications are further divided into separate sub programs for specific types of tests like high voltage insulation tests, AC transfer function measurements, or DC ohmic parameters measurements. Typical window of measurement sub program is presented on Figure 1.

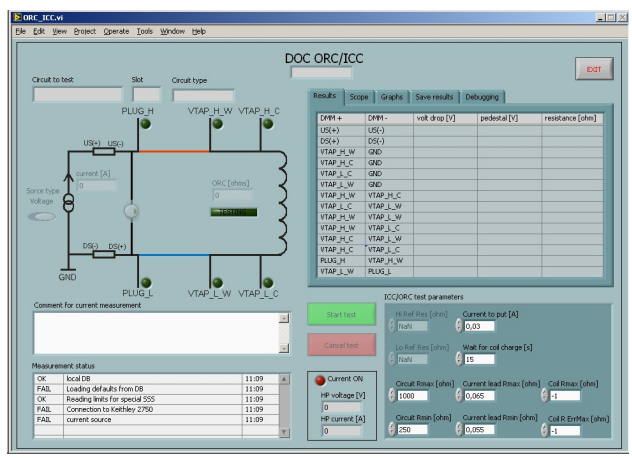


Figure 1: DOC Instrumentation Continuity Check application window.

Each data acquisition application is launched via special panels that help the user to identify the tested circuit and perform proper connections. Figure 2 presents a window of one of such panels.

Panels are designed in a way which makes skipping of the measurement by mistake impossible. Test object lists are ordered by the physical position in the LHC tunnel.

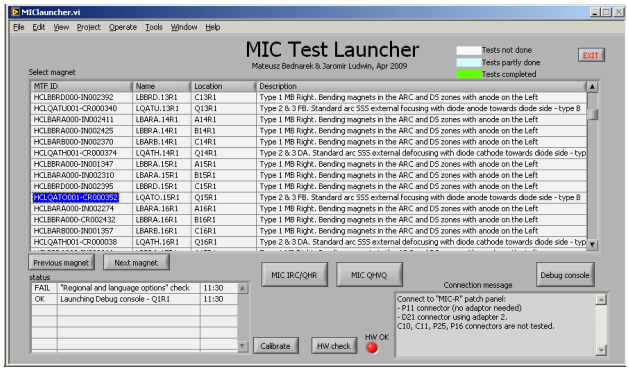


Figure 2: MIC Test Launcher, with list of available magnets to be tested and information about hardware connections that are required to perform tests on a selected magnet.

SYSTEMS' FLEXIBILITY

There were several issues that had to be kept in mind when developing the test applications for the TP4 family:

- There exist many types of circuits with different kinds of diagnostic connectors and various types and numbers of current leads.
- At the development's starting point it was not clear which signals should be measured and which precision was needed.
- A very flexible way of validating the results was needed as the acceptance criteria had to be adjusted together with the measurement campaign progress and more statistics becoming available.

Taking into account those requirements it was decided to create an application which makes use of independently prepared test scripts and validation scripts. The idea of test scripts is presented on Figure 3.

The scripts are stored in the central database. A dedicated tool can be used by a manager in order to create and store a correct script. The script is then copied to the local database of each test bench as soon as their network connection is established.

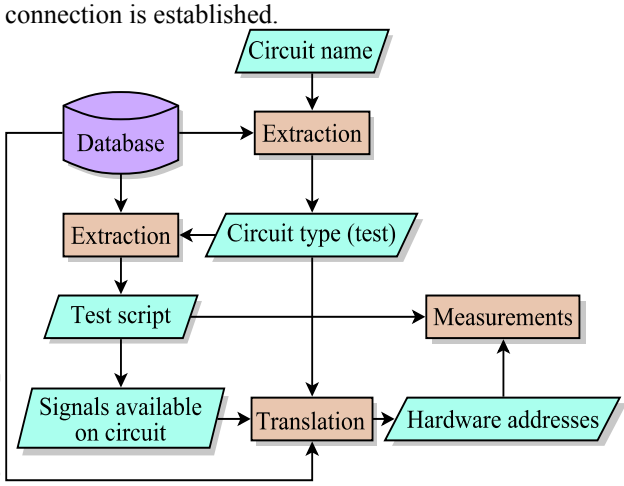


Figure 3: Signal translation in low voltage test applications.

The application determines the type of a given circuit based on the circuit's name and lookup table and extracts the proper test script. Then based on the type, it translates the signal names available on the circuit into the hardware addresses of the measurement system. This translation is used when parsing the test script. Like this no specific information is hard-coded in the source code. Adding tests, measurements and introducing new connector pin-outs is as easy as adding new records in the database.

A similar idea was applied to the verification criteria of recorded values. The great complexity of the cabling requires taking different criteria for each circuit type. As already mentioned the acceptance thresholds were often modified in order to follow the best, growing knowledge about the qualified circuits. Furthermore checking the correctness of electrical connections in the circuit often requires evaluating several mathematical formulas. Therefore the application responsible for the validation process is equipped with a mathematical formula parser.

Figure 4 shows dataflow used during the measurement results verification.

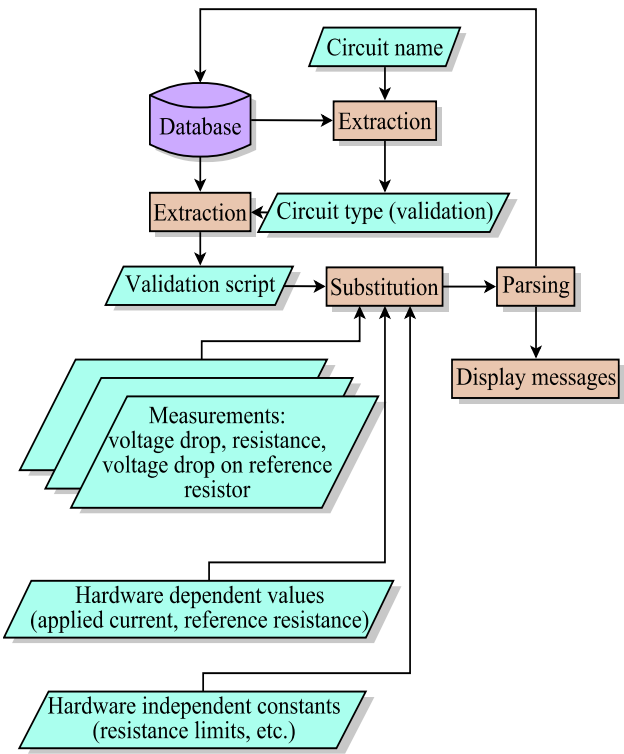


Figure 4: Measurement results validation process.

The most important data stored in the validation script are: left and right value saved in a form of mathematical expression containing test specific variables, comparison operator between left and right value, short information about the circuit property that is being checked and a message to be displayed in case the comparison of left and right values has failed.

TESTS FOLLOW UP AND REPORTING

Each application family has its own set of web pages used to check the current status of tests, analyse results and modify or add nonconformities data. Example of TP4 follow-up page is shown on Figure 5.

Tools	DFB	Circuit	IRC	ICC	TFM	TFM vs GND	HVQ	Flag	Comments
	DFBAD.7R2	RB.A23	OK	OK	OK	OK	OK (2)		**Tests after nQPS cables refurbishment in sector 2-3** 18/01/10: Old QPS HV test in C21R2: 1900V, 2min, 411 GOhm/ 4.72 nA, and C18L3: 1900V, 2min, 205 GOhm/ 9.46 nA. Test successfully repeated after repair, on the 20/01/10.
	DFBAD.7R2	RCD.A23B1	OK	OK	OK	OK	OK		
	DFBAD.7R2	RCD.A23B2	OK (3)	OK	OK	OK	OK		
	DFBAD.7R2	RCO.A23B1	OK	OK	OK	OK	OK		**Tests at cold in sector 2-3 (II)** IRC Failed. Voltage tap resistance EE121 of current lead 1 is too high (4.3 instead of 1.2Ohm). Diagnostic is ongoing. NC 1003638 initiated. Diagnostic showed that the problem is coming from connector P93-1, Pin 1, at the level of PE. A new canon pin replaced the old one (bad contact). Test repeated and passed. NC closed.
	DFBAD.7R2	RCO.A23B2	OK	OK	OK	OK	OK		
	DFBAD.7R2	RCS.A23B1	FAILED (1 of 2)	OK	OK	OK	OK	tests completed	
	DFBAD.7R2	RCS.A23B2	OK	OK	OK (2)	OK	OK		
	DFBAD.7R2	RQ10.R2	OK	OK	OK (3)	OK	OK		**Tests at cold in sector 2-3 (I)** Test is OK.
	DFBAD.7R2	RQ6.R2	FAILED (1 of 2)	OK	OK (3)	OK	OK	tests completed	**Tests at cold in sector 2-3 (I)** IRC test OK. Waiting for cryo condition (level) on Q6 to perform HV test. Cryo condition OK, tests passed.
	DFBAD.7R2	RQ7.R2	OK (2)	OK (2)	OK (6)	repaired (2)	OK	tests completed	
	DFBAD.7R2	RQ8.R2	OK	OK	OK (3)	OK	OK	tests completed	
	DFBAD.7R2	RQ9.R2	OK	OK	OK (3)	OK	OK		**Tests at cold in sector 2-3 (I)** Test is OK.
	DFBAD.7R2	RQD.A23	OK	OK	OK	OK	OK		**Tests at cold in sector 2-3 (III)** ICC test considered as OK. Just above limit, and issue about stability and filter on RQD and

Figure 5: TP4 Follow-up page.

Access to the web pages is controlled via CERN's authentication system. After authorisation the web application is checking user permissions. A certain group of users is allowed to add or change data. All the others are only allowed to view. Any data change made via the web page is logged.

In case of tests where the result is represented as a set of multiple values, the user can export the data in a Comma Separated Values format by clicking the proper link on the test results page. For some measurements, where many identical objects are tested (e.g. quench heaters) web based statistical analysis tools were created.

In addition, a set of LabVIEW report generating programs was created, which can be used to prepare well formatted MS Word documents containing a complete history of the selected circuit within the selected LHC commissioning campaign. Such a document can be then printed and used during an intervention on site.

OTHER TASKS

Having many LabVIEW applications installed on many computers combined with the fact that most of the applications are under constant development it was necessary to create a tool that would help to deploy the new software versions.

The package manager is an application used for installation and update of the ELQA software. It is a batch script that can be started on any MS Windows machine. The script uploads test data to the central database, creates a back-up of the currently installed software, installs the most recent released version and downloads the most recent test parameters. Most of those operations are performed by calling old or freshly downloaded LabVIEW applications with proper command line parameters.

The log of the operation is available for the administrators so that all the errors can be easily tracked.

CONCLUSIONS

The very difficult and complex task of performing the electrical quality assurance of a big prototype machine such as the LHC was successfully accomplished thanks to the described software tool-chain.

The applications are based on a centrally managed database system that automatically propagates the machine layout changes, safely stores the measured data and allows a follow-up in order to avoid accidental omitting of some tests.

The three application families described in this paper are utilising over 500 LabVIEW Virtual Instruments created for the LHC electrical circuits' tests.

FURTHER WORK

Currently the hardware part of the measurement system is being upgraded in view of the upcoming long shut-down in the LHC. As new functionalities and improvements will be added, the software upgrade will follow.

In the future it is planned to add more web based statistical analysis tools. Another planned web application type are tools for easy tracking of circuit parameters change over whole time of LHC operation.

ACKNOWLEDGEMENTS

The authors would like to thank their supervisors for the great opportunity of taking part in the commissioning of the LHC. Also they would like to express their gratitude to the colleagues that supported the development of the system with clever hardware solutions without which the software would not exist. A great number of collaborators from CERN and IFJ-PAN have taken part in the measurements in the LHC tunnel and acted as testers providing the developers with precious comments and hints.

REFERENCES

[1] A. Kotarba et al., "Automatic Measurement System for Verification of the LHC Superconducting Circuits", IPAC'11, San Sebastian, September 2011, TUPS093; http://www.JACoW.org.

[2] D. Bozzini et al., "Automatic system for the D.C. high voltage qualification of the superconducting electrical circuits of the LHC machine", EPAC'08, Genoa, June 2008, WEPD008. p. 2416; http://www.JACoW.org.

[3] A. Kotarba et al., "Electrical quality assurance of the superconducting circuits during LHC machine assembly", EPAC'08, Genoa, June 2008, WEPD016. p. 2440; http://www.JACoW.org.

THE TIMING MASTER FOR THE FAIR ACCELERATOR FACILITY

R.C. Bär, T. Fleck, M. Kreider, S. Mauro
GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany

Abstract

One central design feature of the FAIR accelerator complex is a high level of parallel beam operation, imposing ambitious demands on the timing and management of accelerator cycles. Several linear accelerators, synchrotrons, storage rings and beam lines have to be controlled and reconfigured for each beam production chain on a pulse-to-pulse basis, with cycle lengths ranging from 20 ms to several hours. This implies initialization, synchronization of equipment on the time scale down to the ns level, interdependencies, multiple paths and contingency actions like emergency beam dump scenarios.

The FAIR timing system will be based on White Rabbit [1] network technology, implementing a central Timing Master (TM) unit to orchestrate all machines. The TM is subdivided into separate functional blocks: the Clock Master, which deals with time and clock sources and their distribution over WR, the Management Master, which administrates all WR timing receivers, and the Data Master, which schedules and coordinates machine instructions and broadcasts them over the WR network.

The TM triggers equipment actions based on the transmitted execution time. Since latencies in the low µs range are required, this paper investigates the possibilities of parallelisation in programmable hardware and discusses the benefits to either a distributed or monolithic timing master architecture. The proposed FPGA based TM will meet said timing requirements while providing fast reaction to interlocks and internal events and offers parallel processing of multiple signals and state machines.

FUNCTIONAL BLOCKS

Clock Master

The Clock Master is the topmost White Rabbit timing node in the system. It is the time and clock reference for all connected nodes [1] and is itself connected to a UTC source (GPS) and FAIRs RF clock system, BuTiS. It propagates absolute time and its clock down the layers of the timing network by means of a modified Precision Time Protocol (PTP) and synchronous GigaBit Ethernet.

Management Master

In the Management Master, all administrative tasks of the timing network are concentrated. There are multiple services running in this unit, allowing it to carry out the following functions:

- Dynamic Host Configuration Protocol (DHCP) Server
- Rapid Spanning Tree Protocol (RSTP) Root
- Simple Network Management Protocol (SNMP) Master

A standard DHCP server process is used to assign IP addresses to all nodes present in the timing network.

The SNMP Master is used to poll network switches for information about their current status in order to obtain a global status image of the timing network. It can also be used by administrators to configure said switches.

Since the FAIR Timing Master will broadcast command messages, the RSTP protocol is used to make the routing information for the timing network loop free. This is necessary because a loop in the routing tables would immediately lead to an infinite generation of network traffic.

Data Master

The Data Master for the FAIR accelerator deals with various different tasks. It possesses a high end CPU running an OS for easy interfacing to the control system, compatibility to FAIRs standard libraries and raw processing power as well as a Field Programmable Gate Array (FPGA) for parallelism, deterministic behavior and ultra low IO latency.

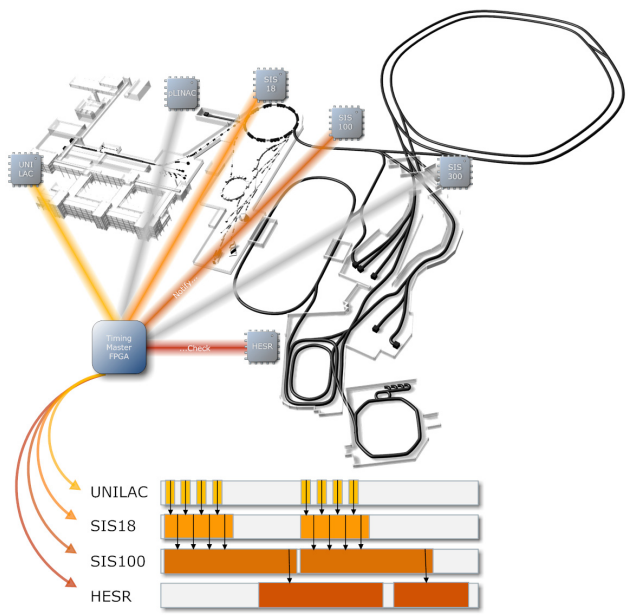


Figure 1: Sequences in a production chain [2].

The Data Master itself subdivided into three parts:

CPU/API Block Its CPU is fed by the LHC Software Architecture (LSA) with machine parameters derived from

physical requirements for beam production chains. These parameters are converted into sequence programs (Fig. 1), compiled and uploaded to the FPGA of the Data master.

FPGA/SoftCPU Cluster These programs are run in parallel on SoftCPU macros residing in the FPGA. They deal with sending out machine events paired with an execution time to WR nodes, reacting to interlocks and mutual synchronisation. At the moment, 32 of these Soft CPUs are foreseen in the Data Master, able to carry out 32 tasks full parallel with IO service times of less than 50ns.

FPGA/Event Concentrator An event concentrator macro in the data master will act as a bridge to the WR network. Its primary functions are aggregation of events into Ethernet Frames and to schedule transmission of these event messages over the timing network so they arrive on time at the respective nodes.

DESIGN CONSIDERATIONS

Why not just a High End CPU?

Modern multi-core CPUs are unsuitable because they cannot they be made deterministic enough for our purpose. This is not inherent in their design but stems from the fact that they need a full blown OS to use most of their features, not to mention RAM management. They also have comparatively slow IO handling. Even with high clock rates and vast amount of processing power at their disposal, their Interrupt Service Request (ISR) times are in the low millisecond range plus additional penalties from IO resource arbitration. While there are approaches which enforce a task based scheduling to introduce determinism, they still suffer heavily from the IO bottleneck of the carrier board.

High end multicore CPU

+ Performance clock for clock
+ Clock rates
– Determinism with (needed) OS
  – multithread scheduler
  – indirect memory management
  – garbage collection
– ISR times range in milliseconds
– IO Bottleneck

➠ Not suitable

Why no Embedded CPU with a Real Time Operating System?

While being considerably faster and more deterministic at IO and interrupt service times than high end CPUs, the ISR of embedded CPUs with real-time OSs still jitter in the microsecond range and show considerable lag under load, as shown in Figure 1.

Table 1: RTOS Latency Measurement Results [3].

	Interrupt Latency (µs)		**Context Switching** (µs)	
	max	**avg ±**	**max**	**avg ±**
Idle System				
RTL	13.5	(1.7 ± 0.2)	33.1	(8.7 ± 0.5)
RTEMS1	14.9	(1.3 ± 0.1)	16.9	(2.3 ± 0.1)
RTEMS	15.1	(1.3 ± 0.1)	16.4	(2.2 ± 0.1)
vxWorks	13.1	(2.0 ± 0.2)	19.0	(3.1 ± 0.3)
Loaded System				
RTL	196.8	(2.1 ± 3.3)	193.9	(11.2 ± 4.5)
RTEMS1	19.2	(2.4 ± 1.7)	213.0	(10.4 ± 12.7)
RTEMS	20.5	(2.9 ± 1.8)	51.3	(3.7 ± 2.0)
vxWorks	25.2	(2.9 ± 1.5)	38.8	(9.5 ± 3.2)

Embedded CPU with Real time OS

+ Fast IO
+ More deterministic
– ISR jitter
– lag peaks under load
– scalability

➠ Not suitable

A multi MCU approach to decreases IOs per MCU to be serviced and processor load would suffer from poor possibilities for process synchronisation between MCUs and lag from resource arbitration.

Why not Pure HDL on FPGAs?

Programmable hardware was the next option to investigate. While being extremely fast and the only system group capable of massive parallel processing, there are also disadvantages to consider. The main problem with HDL modules is a distinct lack of flexibility, since all sequential behavior needs to implemented as state machines. Those can only change their state in reaction to a set of external signals and their own state. This would severely increase the effort for conversion of LSA output and also mean that on the fly changes of functionality are out of the question. However, there is another option when using HDL macros.

Pure HDL circuits

+ completely deterministic
+ ultra low IO latency
– interfacing
– flexibility
– versatility
– design effort for all scenarios

➠ Not suitable

Why SoftCPUs?

SoftCPUs, little blocks of Hardware Description Language (HDL) code resembling embedded CPUs. They can offer almost all of the convenience and flexibility of a real embedded CPU, while also being closely connected to other FPGA circuits with extremely low latency. If there is no necessity for a MMU or the intention to use standard libraries, this is the best compromise between flexibility and ultra low latency design. Multiple instances with their own dedicated memory can speed up the design even more.

SoftCPU Cluster

+ completely deterministic
+ ultra low IO latency
+ interfacing
+ flexibility
+ versatility

➠ Best candidate

ARCHITECTURE

Choice of SoftCPUs

In the course of our work, many SoftCPUs have been evaluated [4]. We decided against closed source variants to keep the possibility of extending the macro with custom instructions. There is about a dozen open source SoftCPUs available today. The main criteria were speed, footprint, availability of toolchains and community/developer support. Our choice was the Lattice Micro32, a 32 Bit RISC processor macro running at about 150 MHz. While being a complete RISC processor with instruction and data caches and an interrupt handler, it has a low FPGA footprint of about 2000 logic cells. Toolchains are available, including a GNU cross compiler for Ansi C. We have also added support for the GNU debugger via JTAG interface. An average modern PFGA can offer around 250000 LUTs, four times more than enough to instantiate a SoftCPU cluster with 32 modules. Internal FPGA memory is much more likely to be the limiting factor, though. High end FPGAs contain >2 MB of internal memory. Assuming 2.5 MB, each of the SoftCPUs would get around 80kB of memory. This is more than sufficient for caches and program execution. The GNU debugger claimed about 1 kB of memory on our LM32 testbed.

Using Multiple SoftCPUs

With a One-SoftCPU-per-task policy, their IO handlers can be blocking and are not forced to use interrupts, eliminating time consuming context changes. Each SoftCPU has its own little memory controller, which completely eliminates resource arbitration lag. For fastest synchronisation of their programs, an $n * n$ sync matrix will also be implemented.

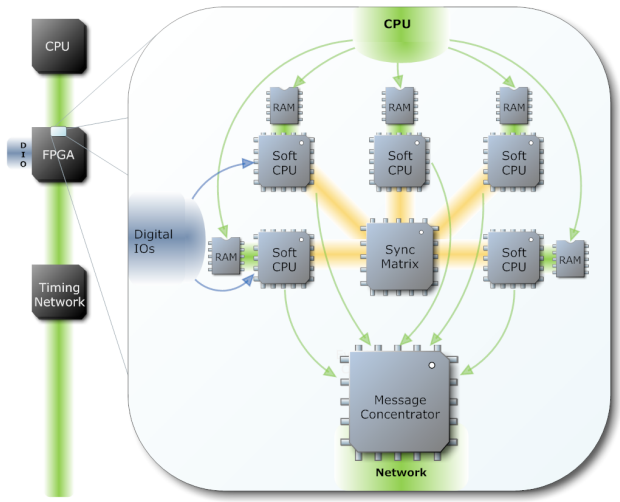


Figure 2: FPGA based SoftCPU Cluster [2].

CONCLUSION

After evaluation of timing requirements [2] and CPU, MCU and FPGA system properties, it showed that a mixed CPU/SoftCPU approach for the Data Master would be the best choice for FAIR. It is a very fast, flexible, scalable, easily extendable and future proof solution.

OUTLOOK

In the course of 2012, the design will be subjected to a real world test scenario. A first prototype of the FAIR Timing Master will be used in the testing and commissioning of the first machine module to be deployed for FAIR, the antiproton linear accelerator. Productive systems are planned to be put into service at GSI/FAIR in 2016.

REFERENCES

[1] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, G. Gaderer, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet", IEEE Precision Clock Synchronization for Measurement, Control and Communication 2009, pp1-5, Brescia, October 2009.

[2] M. Kreider, "The FAIR Timing Master: A Discussion of Performance Requirements and Architectures for a High-precision Timing System", THCHMUST06, ICALEPS'11, Grenoble, France, 2011.

[3] T. Straumann, "Open source real-time operating systems overview", ICALEPS'01, San Jose, USA, 2001.

[4] W.W. Terpstra, "The Case for Soft-CPUs in Accelerator Control Systems", 2011, ICALEPS'11, THCHMUST05, ICALEPS'11, Grenoble, France, 2011.

TIMING SYSTEM OF THE TAIWAN PHOTON SOURCE

C.Y. Wu, Jenny Chen, Y.S. Cheng, P.C. Chiu, Demi Lee, C.Y. Liao, K.B. Liu, K.T. Hsu, K.H. Hu, C.H. Kuo,
NSRRC, Hsinchu 30076, Taiwan

Abstract

The timing system of the Taiwan Photon Source provides synchronization for electron gun, modulators of linac, pulse magnet power supplies, booster power supply ramp trigger, bucket addressing of storage ring, diagnostic equipments, beamline gating signal for top-up injection and the time-resolved experiments. The system is based on event distribution system that broadcasts the timing events over optic fiber network, and decodes and processes them at the timing event receivers. The system supports uplink functionality which will be used for the fast interlock system to distribute signals like beam dump and post-mortem trigger with less than 10 μsec response time. The hardware of event system is based on 6U CompactPCI (cPCI) form factor and available in 2011. Implementation of the software is preceding. Timing solutions for the TPS project will be summarized in the following paragraphs.

INTRODUCTION

The TPS is the latest generation synchrotron light source. Event based timing system is applied for TPS due to its high performance and flexibility which has been already verified in many advanced light sources [1]. The EVG/EVR modules for TPS project are available in December of 2010. Efforts to setup the test system were done in the first quarter of 2011. The prototype system had already applied for the TPS 150 MeV linear accelerator commissioning and acceptance during the second quarter of 2011 with highly success. Developing various software support for event system in on going. Preliminary testing of TPS timing system is on going [2]. All development will be finished before the installation of the whole event system which is scheduled in 2013.

INFRASTURCTURE

The TPS timing system is an event based system. A central event generator (EVG) generates events from an internal sequencer and external sources [3]. These events are distributed over optic fiber links to multiple event receivers (EVRs) [4]. The EVRs, which are located in the control system interface layer, decode the events referred to as hardware triggers or software interrupts. For the linac, the decoded events are further encoded by a gun transmitter and sent over a fiber link to the gun high voltage platform. The external event sources include PPS signal which is locked to GPS, AC mains 60 Hz trigger, post-mortem trigger after beam loss and machine protection system trip. The event clock is derived from the 499.654 MHz master oscillator so that it is locked to change in the RF frequency. Fig. 1 shows an overview of complete TPS timing system and interconnections between its basic components. The master oscillator can be used as an external reference from a GPS disciplined Rubidium 10 MHz clock. Fig. 2 shows TPS timing modules include EVG, EVR, EVRTG and linac gun trigger receiver. The major part of the event system is realized as a 6 U cPCI form factor to compliant control hardware standard for accelerator controls at Phase I.

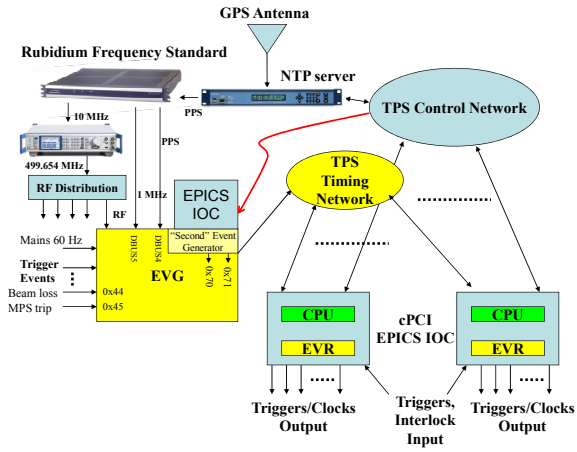


Figure 1: Block diagram of the TPS event system.

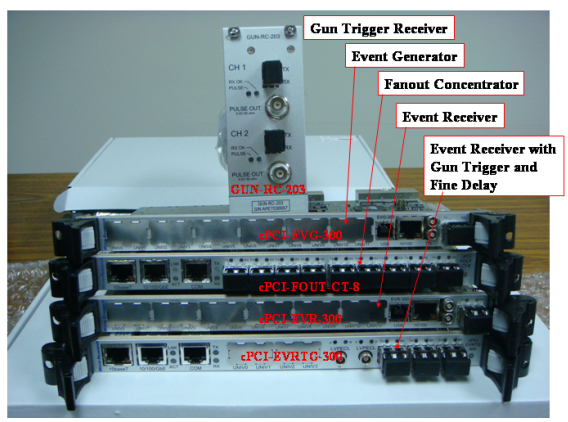


Figure 2: TPS timing modules.

Event Generator

The cPCI-EVG-300 generates event frames consisting of an 8-bit event code and an 8-bit distributed data bus, at a rate of 125 MEvents/sec. Events can originate from external trigger events, sequence RAM, software events and events received from an upstream event generator. Events from different sources have different priorities which are resolved in a priority encoder. A block of RAM is used to store a sequence of events. In cPCI-EVG-300 the input RF clock (499.654 MHz) is divided by 4 to

generate the event clock for the TPS timing system. Therefore, the resolution of timing event is 8 ns.

Event Receiver

The cPCI-EVR-300 and the cPCI-EVRTG-300 are the current available versions of 6U cPCI EVR. The cPCI-EVR-300 recovers the event stream and splits the event frame into the 8-bit event code and the 8-bit distributed data bus. The decoded events are mapped through RAM on to: trigger twelve pulse generators with programmable delay and width (32-bit prescaler from the event clock, 32-bit delay and 32-bit width) or set/reset twelve pulse generators. This EVR provides 3 programmable 16 bit prescaler from the event clock. Six UNIV I/O slots provide twelve front panel outputs can be mapped to any output such as each pulse generator output, prescaler and distributed bus bit. The cPCI-EVRTG-300 has eight channels; it includes two UNIV I/O slots support up to four various output and input, two LVPECL outputs and two SFP outputs. It is embedded jitter cleaner to deliver low jitter functionality of the EVR.

E-Gun Driver

The cPCI-EVRTG-300 has two SFP ports that can generate modulated optical signals that can be decoded by the GUN-RC-203 for linac gun driver trigger. The two SFP ports share an external inhibit signal. The GUN-RC-203 consists of two channels to provide single-bunch and multi-bunch injection respectively. It is realized as a cPCI-EVRTG-300 in the linac timing crate and a GUN-RC-203 placed on the gun HV platform. The fine programmable delay is also available and allowed to adjust the triggering position with a resolution of 10 ps over a range of 10 ns. The GUN-TX-203 mode has been designed to operate output pulse delayed by 1/4, 2/4 and 3/4 event clock period (~2, 4, 6 ns). The new module GUN-RC-300 will support arbitrary pattern generation functionality.

TIMING DISTRIBUTION

The distribution of the event stream will be a fiber optics with tree structure combined with single mode fiber (SMF) as well as multi-mode fiber (MMF) from cost points of view. The long distance link from the timing master to equipments area (Control Instruments Area, CIA, total 24 CIAs around the TPS storage ring and booster synchrotron) will be by SMF, while in CIA to EVR module will be OM3 MMF. Based upon experiences of several running light source, there will several tens of picoseconds drift in the year around due to ambient temperature change of the fiber distribution route. Several tens of picoseconds long-term drift might not cause a problem for most of applications. Drift compensation scheme by sensing fiber which lay around the facility was proposed to measure the long-term drift and make compensation globally. However, this might a correct that the timing fiber link route different path, which might have different environment, the global compensation cannot eliminate the drift for all links. Compensation for each link is standard approach for the RF reference distribution but not for event system yet. To deal with some timing station which need small time drift, the fiber link can replace the phase stabilized optical fiber (PSOF), and PSOF has a propagation delay temperature coefficient roughly 15 times better than standard optical fibers. The scheme of the fiber cabling is still in planning phase. The fiber cabling scheme will be decided in 2012.

TIMESTAMP

The Event System provides a global timebase to attach timestamps to collected data and performed actions. The time stamping system consists of a 32-bit timestamp event counter and a 32-bit second counter.

The timestamp clock and timestamp reset clock are assumed to be raising edge aligned. In the EVG the timestamp reset clock is sampled with the falling edge of the timestamp clock. In the EVR the reset is synchronized with the timestamp clock.

The two seconds counter events are used to shift in a 32-bit second value between every timestamp reset events. In the EVR the value of the seconds shift register is transferred to the seconds counter at the same time the higher running part of the timestamp counter is reset.

The distributed bus event inputs of EVG can transmit timestamp clock and timestamp reset clock independently through the distributed bus event enable register.

EVG will automatically increment and send out two seconds counter events. Using this feature requires the two externally clocks, and the events 0x70 and 0x71 get generated automatically. The timestamp generation of EVG is shown in Fig. 3.

After the rising edge of 1 PPS on DBUS4, the internal seconds counter is incremented and the 32 bit binary value is sent out LSB first as 32 events 0x70 and 0x71. Host CPU can acquire UTC second from NTP server and set second register, load to second counter and enable timestamp generator by on-demand (system boot, cold start).

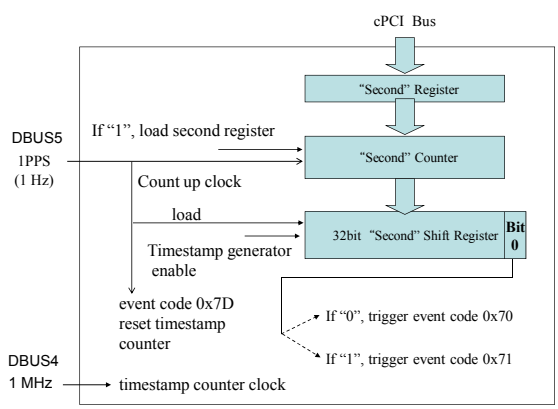


Figure 3: Timestamp generation at EVG.

The timestamp event counter either counts received timestamp counter clock events or runs freely with a clock derived from the event clock. The event counter is

also able to run on a clock provided on a distributed bus bit. The timestamp event counter is cleared at the next event counter rising clock edge after receiving a timestamp counter reset event. The seconds counter is updated serially by loading zeros and ones into a shift register MSB first. The seconds register is updated from the shift register at the same time the timestamp event counter is reset. Fig. 4 shows block diagram of timestamp logic at EVR.

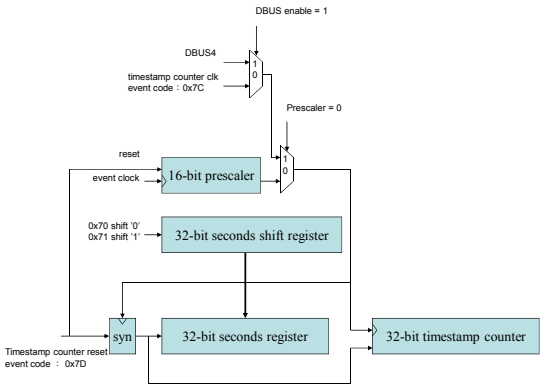


Figure 4: Timestamp logic at EVR.

TIMING SYSTEM DIAGNOSTICS

In normal operation, the rising edges of booster clock, storage ring clock and coincide clock should be lock. Machine clocks are monitored using an oscilloscope. EPICS support for the oscilloscope is prepared. Simple pattern analysis software (Matlab scripts) will analyze the relationship of clocks. If clocks out of lock, it will alarm and stop the system. A time to digital converter (TDC) measure timing difference amount clocks is another auxiliary supports. The wiring schematic is given in Fig. 5. The TDC module will analyze the positions of the clock edges and measures the mismatch. The oscilloscope will display machine clocks waveform for operation. If the machine clock mismatch exceeds 10 ns, then a SYN-FAIL signal will be sent to the machine protection system and notify operators.

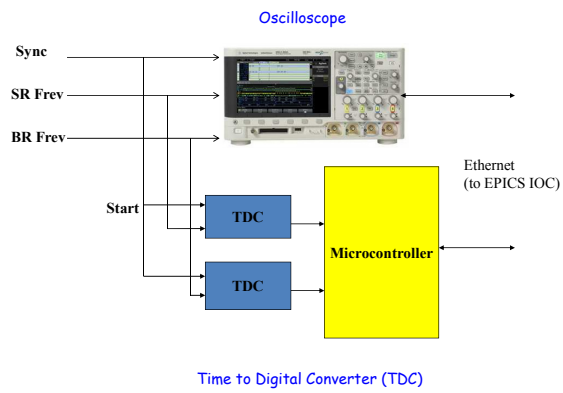


Figure 5: Timing system healthy monitoring.

Based upon the reports of several light sources, there are several tens of psec drift around the year. This is acceptable for most of the accelerators timing and users timing requirement. However, to measure the timing drift of specific links, several approaches can be used.

1. Measure the RF clock versus machine clock edge at EVR site. It needs a stable RF reference available at site.
2. Measured an environment sense fiber around the ring back to the timing master. The difficult is that each link might expose different route on the cable tray. There are some local heating sources on the tray due to the nearby power cables.

TIMING INTERFACE FOR BEAMLINE

The timing interface was developed to provide hardware signals for synchronization of beamline equipment with the electron bunch structure in the storage ring and with the process of Top-up operation. Signals at each experiment station include RF clock, machine clock, top-up injection gating. Top-up gating signals include long gate over whole top up injection cycle and short gate over top up for single injection cycle and few unassigned channels that can be configured to user requirements. For the beamline need high precision timing to synchronize their laser system or data acquisition system, sub-ps RF link will be available.

CONFIGURATION TOOLS

The timing system is integrated into the TPS EPICS control environment. The EVG configuration pages define the options of cPCI-EVG-300. These include configuration of the EVG operating mode, selection of RF and AC divider, definition of multiplexed counter, optional transmission of software events, enable of event trigger inputs and specify event code and timestamp into the sequence RAMs. The EVR configuration pages configure the options for the cPCI-EVR-300, such as pulse delay, width and polarity, front panel output assignments and distributed bus enable and event decoding mapping RAMs. Applications to control the timing system are built with the usual EPICS tools for databases and EDM for user interface.

DOWNLINK AND UPLINK FUNCTIONALITY

The event system not only distributes events to EVRs but also deliver event uplink functionality by the help of the fanout concentrator.

Measurement of the global response time for the TPS timing systems uplink and downlink is presented in Fig. 6 for two stage fanout concentrator and 310 m single mode fiber. This event code 0x45 is used for indication of MPS-TRIP event. The event is stimulated by an external trigger produced by an EVR2. According to the oscilloscope pattern the EVR2 transmitted event by MPS-TRIP trigger and EVR1 decoded event pulse are shifted with respect to each other by 2.45 μs. The global response time is 4.42 μs which consists of EVG, EVR1, EVR2, three fan-out

concentrators processing times and propagation delay time along timing system fiber network (Fig. 6).

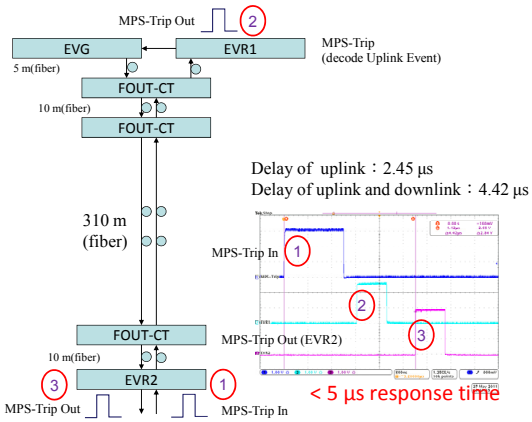


Figure 6: Timing measurement of uplink and downlink with 310 m fiber and three stages of fan-out concentrator.

INJECTION SEQUENCE CONTROL

To accommodate various operation and injection scenarios of the TPS storage ring and booster synchrotron, sequence control plays a crucial role. The sequence is stored at sequence RAM of the EVG module. Adopting Matlab scripting for the sequence control of the TPS accelerator is considered seriously due to expertise of available manpower and high productivities compare to another tools. The timing sequence control is also considered to use Matlab scripts embedded in the timing master EPICS IOC which host the event generator module. Since the TPS machine repetitive rate is 3 Hz, the booster power supply ramping waveform is 3 Hz sinusoidal wave and T-ZERO defined at 0 msec. The storage ring extraction time is T-ZERO+166.66 msec. There are sufficient time for sequence RAM management (start/stop, replace contents of sequence RAM locations …etc.). The scripts will access (read and set) the EVG related EPICS PVs by LabCA. So, the sequence control OPI can set parameters related to the sequence and execute by the sequence control scripts running in the timing master EPICS IOC.

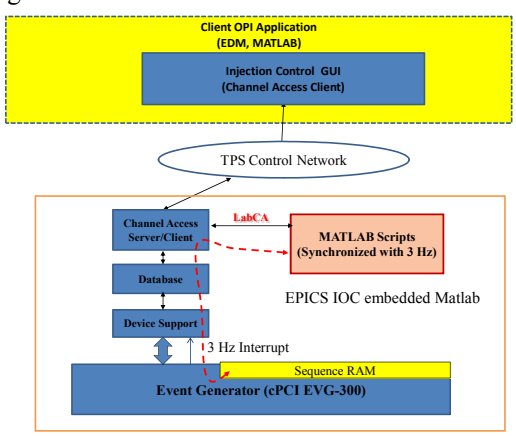


Figure 7: Block diagram of proposed timing sequence control.

The Matlab scripts for injection sequence control detect sequence RAM interrupt, check injection conditions or operation modes and replaces sequence RAM contents. The Block diagram of timing sequence control is represented in Fig. 7. All parameters for the machine operation modes and parameters will be designed as specific EPICS PV, such as bucket address, repeat cycle, top-up injection, decay mode and etc. Preliminary operation modes supported includes:

a. E-gun pulser: multi-bunch and single bunch.
b. Multi-bunch width adjust.
c. Booster magnet cycle before beam trigger.
d. Storage ring septum cycle before beam kicker fired.
e. Next injection bucket address.
f. Top-up-mode injection.
g. Fill pattern management.
h. Inject to target current.
i. Individual element trigger for component test.
j. Booster + linac operation only.
k. Linac operation only.
l. Auto stop injection when safety interlock actuated, stored current exceeds target value.
m. …etc.

SUMMARY

The event system for the TPS project is being implementation stage. The prototype has been deployed in early 2011 for linac acceptance test. Configuration tools have been developed and revised continually. Preliminary sequence control is in proceeding. Timestamp function of event system will be tested.

ACKNOWLEDGEMENT

The authors thank help from many experts on the timing system especially Yuri Chernousko of the DLS, Kazuo Furukawa of the KEK, Timo Korhonen of the PSI/SLS, and Jukka Pietarinen of the MRF.

REFERENCE

[1] Y. Chernousko, et al., “Review of the Diamond Light Source Timing System”, Proc. of RuPAC-2010, Protvino, Russia. pp. 144-146.
[2] C.Y. Wu, et al., “Preliminary Testing of TPS Timing System”, IPAC Proceedings, paper ID: MOPO041, September 2011.
[3] Micro-Research Finland Oy, “Event Generator”, Document: EVG-MRM-0005, 19 September 2011.
[4] Micro-Research Finland Oy, “Event Receiver – Technical Reference”, Document: EVR-MRM-003, 7 April 2011.

NSLS-II BOOSTER TIMING SYSTEM

P. Cheblakov, S. Karnaev, BINP, Novosibirsk, Russia
Joe De Long, BNL, Upton, NY 11973, USA

Abstract

NSLS-II light source includes the main storage ring with beam lines and injection part consisting of 200 MeV linac, a full-energy 3 GeV booster synchrotron and two transport lines.

The booster timing system is a part of NSLS-II timing system which uses hardware from MicroResearch Finland: Event Generator (EVG) [1] and Event Receivers (EVRs) [2,3]. The booster timing is based on the events coming from NSLS-II EVG: "Pre-Injection", "Injection", "Pre-Extraction", "Extraction". These events are referenced to the selected RF bucket of the storage ring and correspond to the first RF bucket of the booster.

EVRs provide triggers both for the injection and the extraction pulse devices. EVRs also provide the timing of booster cycle operation and generation of events for cycle-to-cycle updates of pulsed and ramping parameters, and synchronization of the booster beam instrumentation devices.

This paper describes the final design of the booster timing system. The timing system functional diagrams and block diagram are presented.

INTRODUCTION

The NSLS-II timing system is based on solutions and experience of the Diamond Light Source timing system [4,5] and uses events coming through fiber optic from the central EVG. The feature that simplifies the booster timing and beam injection from the booster to the storage ring is the fact that ratio of their perimeters is 5. It allows the use of two groups of events referenced to one selected RF bucket of the storage ring. The first group ("Pre-Injection" and "Injection") provides injection of the beam into the booster. "Pre-Injection" event sets the first RF bucket for the booster. All other events coming from the EVG are referenced to this bucket automatically. The second group of events coming in 300 ms after the first group ("Pre-Extraction", "Extraction") provides extraction of the accelerated beam from the booster and injection to the storage ring.

With a help of EVRs, each group of events initiates a set of triggers for triggering of pulsed devices, control and measuring electronics, and beam diagnostics systems. EVRs provide two scales for triggering. The first scale is for triggering the pulsed septums and the bump magnets in millisecond range and uses TTL outputs of EVRs. The second scale corresponds to triggering of the kickers in microsecond range and uses CML outputs. Also EVRs provide revolution frequency and 10 kHz clocks for control and measurement equipment.

BOOSTER TIMING SYSTEM STRUCTURE

A block diagram of the booster timing system is presented in Fig.1. The booster timing system receives events from the central EVG. These events are distributed over fiber optic links (blue lines) and fan-out modules to EVRs.

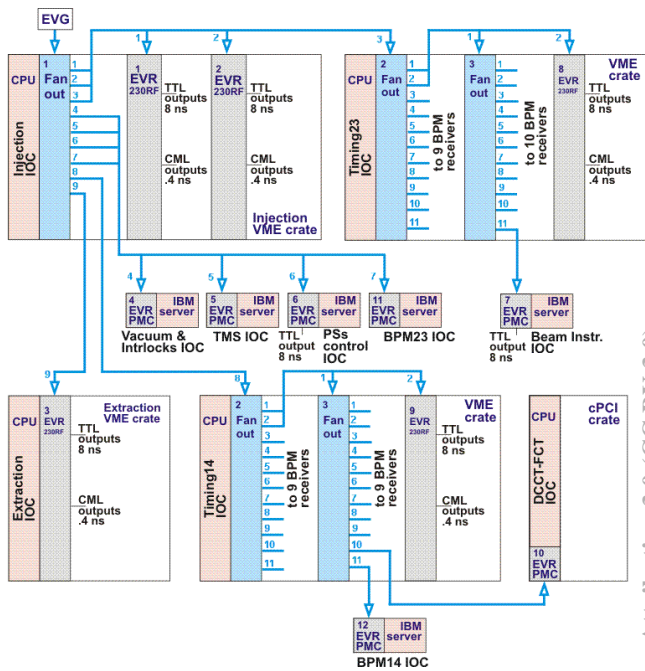


Figure 1: Block-diagram of the booster timing system.

Three EVR-230RF are located in two VME crates: two EVRs are used for injection and one - for extraction. These EVRs provide hardware triggers for injection and extraction equipment.

Also two EVR-230RF are used for synchronization of BPM receivers and other beam diagnostics equipment. These EVRs are placed in VME crates in diagnostic racks and are used for generation of the booster revolution frequency signal and 10 kHz signal for clocking of power supply controllers (PSCs). One EVR generates triggers for cameras in order to provide synchronization with the beam passing.

Six PMC EVRs are placed in six IBM servers. One PMC EVR generates triggers for three chassis with PSCs for providing a synchronous start of the ramp power supplies.

One PMC EVR is placed in cPCI controller for triggering beam diagnostics ADCs which are placed in cPCI crate.

TIMING EVENTS

The time of the booster operation cycle is 1 sec. Two beam injections with a 100-msec interval are proposed to increase the accelerated beam current. The ramp time is 300 msec. A simplified diagram of the booster operational cycle and timing events is presented in Fig. 2. The red curve shows a current value in the bending magnets.

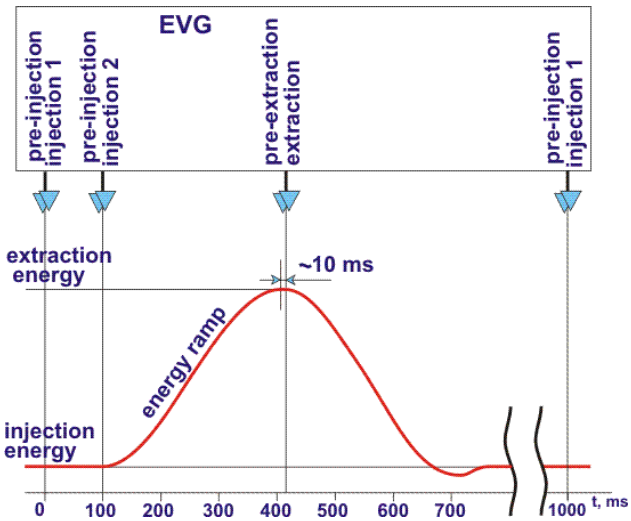


Figure 2: Simplified timing diagram of the booster cycle.

The EVG will provide six events referenced to the selected RF bucket of the storage ring for reference of the booster control timing to the NSLS-II beam:

- Two “Pre-Injection” events spaced 100 msec apart.
- Two "Injection" events coming after 50 turns in the storage ring (132 μs) after "Pre-Injection".
- "Pre-Extraction" event coming in 300 msec after “Pre-Injection” event.
- "Extraction" event coming after 350 turns in the storage ring (925 μs) after "Pre-Extraction" event.

The set of the events listed above provides synchronization between the booster and storage ring bunches.

BEAM INJECTION TIMING

Injection from the linac to the booster is repeated twice with a 100-msec interval in each cycle. A detailed diagram of injection triggering signals from EVRs is shown in Fig. 3.

The following triggering signals are required to control the injection pulsed devices:

- BIT1 (Booster Injection Trigger) - start of the table processing in all Power Supply Controllers (PSCs) [6] (EVR TTL output).
- BIT2 – start of volt-second digitizer for measurement of a pulsed magnetic field value (EVR TTL output).
- BIT3 – start of injection septum current wave form digitizer for digitizing the injection septum pulse current shape (EVR TTL output).
- BIT4 - start of the injection septum (EVR TTL output).
- BIT5-BIT8 – start of injection kickers (EVR CML outputs).
- BIT9 – start of injection kickers current wave form digitizer for digitizing the injection kicker pulse current shape (EVR TTL output).
- BIT10 - stop of the volt-second digitizer (EVR TTL output).

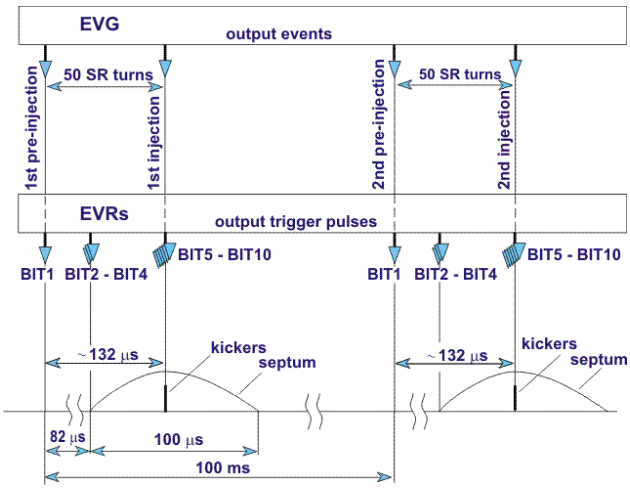


Figure 3: Injection timing diagram.

Both injections use the same triggering signals, but some signals (BIT2, BIT3, BIT9, BIT10) come from different EVR TTL outputs: one output is used for the first injection, another - for the second injection. This allows us to divide measurement channels for the first and the second injections.

Also 10 kHz frequency will be provided from EVR for clocking of PSCs. This signal will be sent from EVR to all PSCs via CML fan-out modules.

BEAM EXTRACTION TIMING

Extraction from the booster is initialized by "Pre-Extraction" event coming 300 msec after “Pre-injection” event. At this moment the beam is accelerated up to 3 GeV and is at the energy flat to provide stabilization of beam parameters during 10 msec. After coming of "Pre-Extraction" event, pulsed bump magnets deflect the beam orbit close to the extraction septum, then, at the moment of maximum deflection, the septum PSs and kicker PSs start. The diagram of extraction procedure is shown in Fig. 4.

The following triggering signals are used for extraction procedure:

- BET1 (Booster Extraction Trigger) - start of extraction Volt-Second Digitizer (VSD) for measurement of bump and septum pulsed magnetic field values at the extraction moment (EVR TTL output).
- BET2 – start of extraction current Wave Form Digitizers (WFD) for digitizing the extraction bump

and septum magnets pulse current shapes (EVR TTL output).

- BET3 – start of bump magnets (EVR TTL output).
- BET4 - start of the extraction septum (EVR TTL output).
- BET5, BET6 – start of extraction kickers (EVR CML outputs).
- BET7 – start of extraction kicker current fast Wave Form Digitizer (EVR TTL output).
- BET8 - stop of the volt-second digitizer (EVR TTL output).

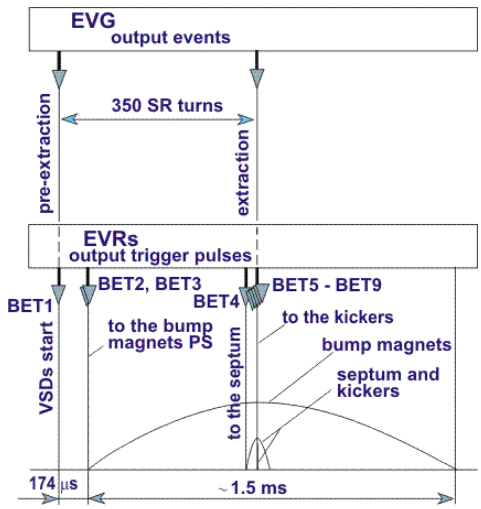


Figure 4: An extraction timing diagram.

BEAM INSTRUMENTATION TIMING

Several triggers are required for the beam instrumentation timing:

- BDT1 (Booster Diagnostics Trigger) - start of high speed digitizer DC222 for FCT (EVR TTL output).
- BDT2 – start of a high resolution digitizer for measurement of signal from DCCT (EVR TTL output).
- BDT3 – start of Tune Measurement System (TMS) controller (EVR TTL output).
- BDT4-BDT9 - start of beam flag cameras (EVR TTL outputs).
- BDT10, BDT11 – start of SR Monitor (SLM) cameras (EVR TTL outputs).

The first "Pre-Injection" event will be used for triggering of on-board EVRs in each of 37 BPM receivers. For this purpose four fan-outs are used in two VME crates (see Fig. 1).

The booster revolution frequency signal will be provided from EVR for Beam Position Monitor receivers and TMS clocking. These signals will be sent from EVR via CML fan-outs.

SOFTWARE

The timing system is integrated into the EPICS based control system through EPICS records. The EVR .db records configure next parameters of the EVR: pulse delay, pulse width, pulse polarity, output assignment, etc. An example of application proposed to be used for control of test stand timing is built with the use of CSS (see the application screen in Fig. 5).

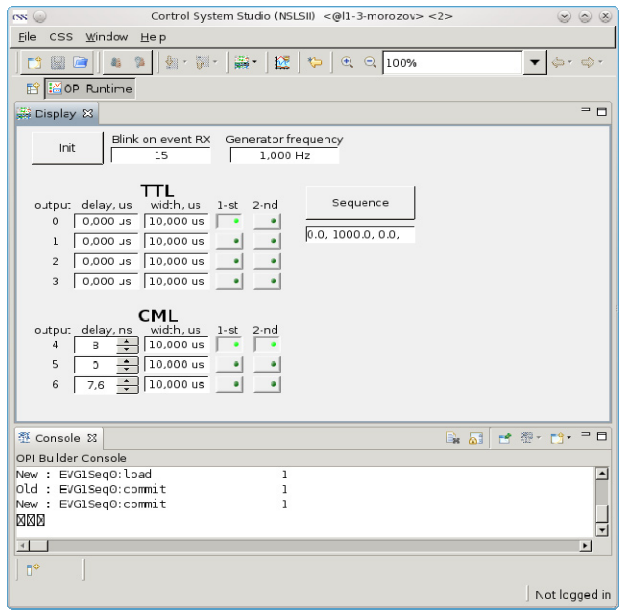


Figure 5: Screen for the EVR control.

SUMMARY

The detailed design of the NSLS-II booster timing system is finished.

The timing system based on the standard equipment provides all required triggers and signals for the booster PSs and electronics.

Some part of electronics now is being used at power supplies test stand.

Software is based on EPICS, now it is under development, its completion is planned before the start of the booster commissioning in October, 2012.

REFERENCES

[1] http://www mrf fi/index php/vme-products/74-vme-event-generator-vme-evg-230.

[2] http://www mrf fi/index php/vme products/75 vme - event-receiver-rf-vme-evr-230rf.

[3] http://www.mrf.fi/index.php/pmc-products.

[4] Y. Chernousko, et al., "Review of The Diamond Light Source Timing System", Proceedings of RuPAC-2010, Protvino, Russia, http://accelconf.web.cern.ch /accelconf/r10/papers/wechb02.pdf.

[5] Y. Chernousko, et al., "Metrological Testing of DLS Timing System", Proceedings of ICALEPCS07, Knoxville, Tennessee, USA, http://accelconf.web. cern.ch/accelconf/ica07/PAPERS/WPPB23.PDF.

[6] P. Cheblakov, et al., "NSLS-II booster power supplies control", talk at this conference, WEPMS020.

NETWORK ON CHIP MASTER CONTROL BOARD FOR NEUTRON ACQUISITION

E. Ruiz-Martinez*, T. Mary, P. Mutti, J. Ratel and F. Rey. Institut Laue-Langevin, Grenoble, France

Abstract

The acquisition master control board is designed to assemble the various acquisition modes in use at the Institut Laue-Langevin (ILL). The main goal is to make the card common for all the ILL's instruments in a simple, modular and open way, giving the possibility to add new functionalities in order to follow the evolving demand. It has been necessary to define a central element to provide synchronization to the rest of the units. The backbone of the proposed acquisition control system is the denominated master acquisition board. The master board consists on a VME64X configurable high density I/O connection carrier board based on the latest Xilinx Virtex-6T FPGA. The internal architecture of the FPGA is designed as a Network on Chip (NoC) approach. The complete system also includes a display board and *n* histogram modules for live display of the data from the detectors.

INTRODUCTION

The master board centralizes the timing information and the synchronization signals used in the hardware management of the data acquisition of the instrument. At the ILL the standard acquisition system common to most of the instruments is based on a VME bus solution, where all the boards become slaves of one master card.

The present article reports on the development of a master board using a general-purpose carrier card for use in future applications. The MPC 1200 carrier board has been developed by collaboration between IOxOS Technologies and the Instrument Control Service (SCI) at the ILL. The document will start with the description of the network on chip approach using a Virtex 6 FPGA. The MPC 1200 carrier board section describes the multiple features, the IO connections and the devices present on the board. The MPC front-end section summarizes the MPF modules used in the carrier board and the design of the front panel used within the master acquisition application. The implementation part explains the main blocks employed in the master acquisition and the development kit provided in the TOSCA II architecture. The discussion will end with the main conclusions gathered in the development of the project.

NETWORK ON CHIP

The standard shared bus architecture, widely used since years, is all based on master/slave interactions. A master reads or writes one or several slave resources, which can be IO registers and/or memory locations. Several new trends have been introduced like for instance cache memory, optimized burst transactions, split transactions with un-compelled request and completion, but without modifying fundamentally the initial principle. These standard shared buses have two main inconvenient:

- the physical media interconnecting the masters with the slaves shall be shared. In order to allocate the media to the master an arbitration mechanism is needed with a direct impact on the latency and the bandwidth performance.
- the physical media size (number of pins) is limited in construction. It is difficult to built interfaces with thousands of IO.

The TOSCA II architecture has been defined to support the shared bus architecture but with optimal latency and bandwidth performance. The implementation has been done in new high performance Xilinx Virtex-6T FPGA. The interconnection between the masters and the slaves are no more implemented with a shared media but with a full mesh switch where the master owns a direct connection to every slave. This approach presents the following advantages.

- due to point to point connection the bandwidth performance is not shared and can be specifically defined by the master and the slave implementation.
- due to individual connection between masters and slaves, the latency is fully under the control of the slave and no more related to the media sharing.

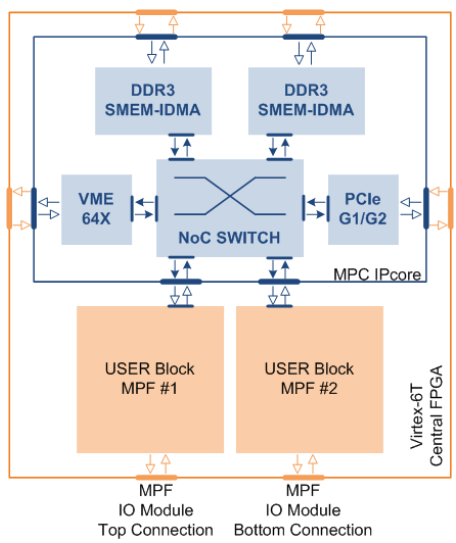


Figure 1: Network on chip architecture.

The NoC approach, see Figure 1, is possible thanks to the fact that the targeted FPGA has enough resources (DPRAM and LE) and that there is no limitation in the number of connections inside the FPGA routing. Another key feature of the TOSCA II Architecture is based on the

*ruizmartinez@ill.fr

full availability of the interface IP core, at VHDL source level for all the key functions (PCI, PCI Express, DDR2/DDR3 controllers, DMA engines, switch infrastructure) [1].

The XILINX Virtex-6T Central FPGA top is organized in a hierarchy level composed of two main blocks:

- the MPC IP core integrating the basic carrier board infrastructure as NoC switch, PCI express EP, VME64x master/slave, the DDR3 shared memories controller with IDMA and the interrupt controller
- two USER blocks, one for each MPF IO connector, fully configurable by the end user.

The MPC IP Core is released in binary format (NGC), along with generic USER block examples and the needed files for the implementation of a fully functional FPGA for the MPC 1200.

MPC 1200 CARRIER BOARD

Compared to other products based on mezzanine cards extension (i.e., FMC or XMC), the MPC 1200 edge-to-edge interconnection solution, see Figure 2, provides full PCB area utilization, direct VME64x front panel access without any restriction on front panel connectors type, and enhanced air cooling capability with standard heat-sink while keeping modularity and versatility.

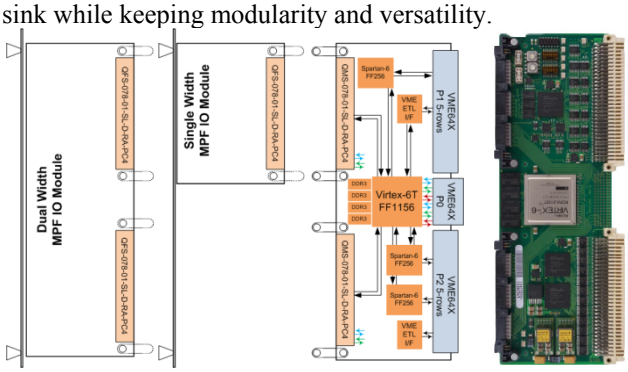


Figure 2: MPC 1200.

The central Virtex-6T FPGA embeds a proprietary IP core providing a complete master/slave VME64x interface. The master/slave VME64x IP core is part of the TOSCA II infrastructure and therefore provides transparent bridging capability such as PCI Express to VME64x [2].

The VME slave interface implements the legacy VME/VME64 addressing configuration with static switches and/or VME64x CR/CSR. The VME slave handles all addressing modes and is fully programmable through an in-going scatter-gather. The VME master interface, driven by the internal FPGA infrastructure supports also all VME addressing mode including 2eVME and 2eSST.

The MPC 1200 integrates on-board DC-DC power supplies for the on-board FPGA and for the MPF IO modules. These DC-DC power supplies are also dimensioned to support the whole Virtex-6T device family. To fully support the Virtex-6T Central FPGA capability, the MPC 1200 integrates three additional Spartan-6 XC6SLX9- FF256 companion devices. Two are dedicated to the VME P2 User IO support and the third one is used for power-up management and FPGA configuration.

The VME P2 user IO (32+32+40 signals on 5-rows connector, see Figure 3) is supported through the two Spartan-6 programmable routing infrastructure providing a 5 V tolerant electrical interface with the P2 connector. The interconnection with the central Virtex-6T FPGA is supported through high-speed (up to 1Gb/s) low level 1.8 V SE or DE signaling.

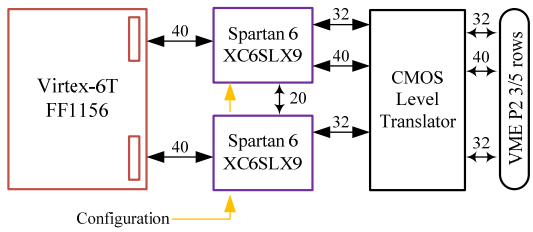


Figure 3: VME64x P2 User IO.

The master acquisition board uses the VME P2 signals to manage the synchronization among the different acquisition and histogram modules. These signals are shared with the boards present in the crate. The input/output direction of the P2 signals is configurable depending on the instrument mode of acquisition.

The FPGA configuration is assured by power on (PON) finite states machine (FSM) implemented in a Spartan-6. The FPGA configuration bit-streams for the Spartan-6 and Virtex-6T are all stored in high-speed serial flash EPROM devices. The FPGA bit-streams can be directly downloaded from the VME slave interface or PCI Express EP.

The carrier board includes as well four 128 Mbit SPI Flash EPROM devices with 4-bit read-out. Four DDR3 memory devices are connected to the Virtex-6T central FPGA. These memory devices can be interfaced as two or four independent DDR3 controller. The DDR3 memory controller IP core provides the complete control of the DDR3 devices and it has been fully integrated in TOSCA II. Other functions such as I2C and SPI interfaces, GPIO management, PON FSM controller, and power supplies monitoring, are integrated within the Virtex-6T central FPGA firmware.

MPF FRONT END

Two Multi-Purpose Front IO (MPF IO) module sizes are supported. The single-width size is derived from the 3U eurocard and the dual-width from the 6U eurocard.

The MPF IO Modules solution has several advantages compared to the FMC extension approach:

- larger PCB size available
- direct access to the front panel without limitation on the IO connectors (no micro connectors required).

- full VME component height allowing installation of heat-sink on critical devices.
- better physical and galvanic isolation for noise sensitive application (i.e. ADC with resolution better then 12-bit).

The MPF IO modules are attached to the MPC 1200 carrier through the SAMTEC QFS/QMS connectors (see Figure 4). The MPC 1200 provides to the MPF modules a full collection of power supplies with up to 4.1 A. (8.2 A on the 1.0 V). For each MPF connector the Virtex-6T IO support is limited to 100 SE (50 DE) signals distributed over 2.5V Virtex-6T IO banks. The use of Spartan-6 devices as data concentrators boosts data bandwidth up to 50 Gb/s [2].

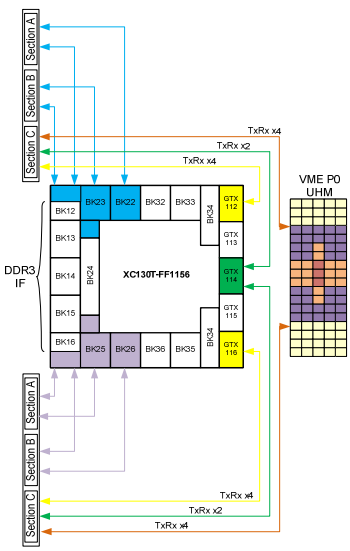


Figure 4: Front panel MPF IO connection.

The front end of the master acquisition board contains the signals coming from external modules involved in the acquisition procedure, like pulses from a rotating machine or an external clock for synchronization purposes. These signals have been grouped in a customized front panel with 12 one-point LEMO for individual inputs, and a DB26 high-density connector. In future applications the front panel could be modified including a processor, flash ADC digitizers or high speed digital connectors for high performance applications [3].

IMPLEMENTATION

The implementation of the master acquisition board has been developed in VHDL using the user area of the TOSCA II architecture. TOSCA II is delivered with full VHDL source code together with a set of test-benches and bus functional models (BFM) to set up a complete VHDL simulation environment for functional verification purposes. The TOSCA II architecture is based on a PCI Express switch centric structure implementing a memory mapped model with segregated I/O space (control plane) and memory space (data plane) [4].

In the master acquisition board each acquisition mode in use at the ILL is developed in a functional and independent block. These blocks are able to manage the synchronization signals, using the front-end and the VME P2 interface, according to the specific needs of the instrument. Depending on the measurement technique adopted, the following acquisition modes are accessible via the master board [5]:

- Simple integral count: available for both single and multi-detectors, it allows a neutron's counting time from 100 *ns* to $(2^{64}\text{-}1)\cdot 10^{8}$ *s* with a time resolution of 100 *ns*. While the gate generated by the master acquisition board remains active all the neutrons arriving at the detector will be counted and the corresponding histograms are stored in the acquisition cards.
- Time of Flight (ToF): the continuous neutron flux produced by the ILL's reactor is pulsed by means of a rotating chopper with frequencies between a few Hz up to 500 Hz. The energy of a detected neutron is calculated starting from the traveling time from the chopper to the detector once the distance of the two elements is known. The master acquisition board is able to manage four independent ToF consisting of up to 2048 variable duration time channels, from 100 *ns* up to 43 *s*. The acquisition card will provide the corresponding histograms where all the detected neutrons are arranged as a function of their arrival time.
- Kinetic: used especially for soft matter studies to trace phenomena with long duration in time, it is very similar to the previous mode. In this case the neutron flux is continuous and the start signal is delivered by an external event. The acquisition card delivers time-slice histograms where the neutrons are arranged according to their arrival time. A combination of ToF and Kinetic mode is also possible when a chopper is available on the instrument. In this case the neutrons in each kinetic time-slice are arranged according to their respective ToF.
- Doppler: two backscattering spectrometers are equipped with a Doppler motion at the monochromator. This last one is moved with a velocity parallel to the reciprocal lattice vector [6] and, as a consequence, the energy of the reflected neutron is modified via a longitudinal Doppler effect. The velocity of the monochromator is varied periodically following a sinusoidal function. The master acquisition board receives via the front-end external synchronization signals corresponding to different velocities. The acquisition card produces a histogram where the neutrons are accumulated into the corresponding equidistant velocity channels.
- Oscillating Radial Collimator (ROC): diffraction experiments on instruments equipped with Position Sensitive Detectors (PSD) suffer from background originated by the sample environment [7]. A specific diffracted beam collimator has been designed to reduce the off-sample parasitic scattering without

perturbing the resolution or the scattering from the sample. This collimator is oscillated through a specific angle with a linear angular velocity. The master acquisition board coordinates the neutron's count depending on the movement of the collimator.

CONCLUSION

As shown in the previous section, the instruments at the Institut Laue-Langevin required a large variety of acquisition modes. This, together with the need of properly synchronize several acquisition cards, has pushed the development of a new master acquisition board. An inappropriate management of all the different external signals connected to the acquisition could cause loses of information during the experiment and/or in the subsequent data analysis. Versatility is a mandatory requisite for the new board since it should be adapted to all the instruments without major changes. The results of all those constraints is a master board design conceived as a VME carrier board with high density I/O connection, leaving the autonomy to use the front-end for custom applications. The board is based on a central Virtex-6 FPGA which embeds a full mesh switch where the master owns a direct connection to every slave, having fully connection with the resources present in the board: PCI express, VME64X master/slave, DDR3 memory controllers and the user area, where the master acquisition application has been implemented.

REFERENCES

[1] J. Bovier, "MPC 1200 Hardware-Software Interface Data", IOXOS technologies internal report, (2009).
[2] J. Bovier, "MPC 1200 – Multi-Purpose IO Carrier VME64X IO Interface", IOXOS technologies, (2011); http://www.ioxos.ch.
[3] T. Mary, F. Rey, J. Ratel, "New acquisition system", Institut Laue-Langevin internal report, (2011).
[4] J. Bovier, "MPC 1200/ IFC 1210 TOSCA II Simple USER Block", IOXOS technologies internal report, (2011).
[5] J. Ratel, "Interface VME Master Acquisition", Institut Laue-Langevin internal report, (2011).
[6] B. Frick, "Neutron and X-Ray Spectroscopy", Part 2, p. 483 (2006).
[7] J. Bouillot and J. Torregrossa, Revue Appl. Phys. 19 (1984).

THE GLOBAL TRIGGER PROCESSOR: A VXS SWITCH MODULE FOR TRIGGERING LARGE SCALE DATA ACQUISITION SYSTEMS

S. Kaneta, C. Cuevas, H. Dong, W. Gu, E. Jastrzembski, N. Nganga, B. Raydo, J. Wilson,
Jefferson Lab, Newport News, VA, U.S.A.

Abstract

The 12 GeV upgrade for Jefferson Lab's Continuous Electron Beam Accelerator Facility requires the development of a new data acquisition system to accommodate the proposed 200 kHz Level 1 trigger rates expected for fixed target experiments at 12 GeV. As part of a suite of trigger electronics comprised of VXS switch and payload modules, the Global Trigger Processor (GTP) will handle up to 32,768 channels of preprocessed trigger data from multiple detector systems that surround the beam target at a system clock rate of 250 MHz. The GTP is configured with user programmable Physics trigger equations and when trigger conditions are satisfied, the GTP will activate the storage of data for subsequent analysis. The GTP features an Altera Stratix IV GX FPGA allowing interface to 16 Sub-System Processor modules via 32 5-Gbps links, DDR2 and flash memory devices, two gigabit Ethernet interfaces using Nios II embedded processors, fiber optic transceivers, and trigger output signals. The GTP's high-bandwidth interconnect with the payload modules in the VXS crate, the Ethernet interface for parameter control, status monitoring, and remote update, and the inherent nature of its FPGA give it the flexibility to be used large variety of tasks and adapt to future needs. This paper details the responsibilities of the GTP, the hardware's role in meeting those requirements, and elements of the VXS architecture that facilitated the design of the trigger system. Also presented will be the current status of development including significant milestones and challenges.

GTP RESPONSIBILITIES

Level 1 Trigger

As part of the 12 GeV trigger upgrade, the GTP is the central processor for the Level 1 trigger in Jefferson Lab's new data acquisition system. It is designed to simultaneously generate up to 32 independent, programmable triggers with a fixed latency from the event occurrence at a 4 ns resolution.

The new system will essentially eliminate trigger dead time by pipelining the detector sampling and trigger decision calculations. The data pipeline replaces long spools of delay cable previously used and instead stores the data and time samples in memory devices on the VXS ADC and TDC payload boards. When triggers are received, the associated data are read out of these buffers for long-term storage.

The trigger calculations will also be pipelined and computed in parallel to maintain continuous availability of triggers. In order to ensure there is no trigger processing dead time, the triggers must be able to fire successively at the 250MHz global clock. The trigger must also reach the front end modules before the buffers run out of space and overwrite data which satisfy trigger conditions. Hardware limitations define a maximum buffer size of 3.2 µs [1].

Synchronization of the trigger data is also critical to ensure triggers are calculated correctly. A clock synchronization signal is used to align the data of all transmitters so that the summation across channels uses samples from the same point in time. It also helps align transceiver channels themselves to remove skew caused by the deserializers. This is done by aligning the reading of the transceiver memory buffers after all channels have received valid data.

Figure 1 shows the flow and concentration of data through the trigger system data path. Each Crate Trigger Processor (CTP) accepts data from its TDC or flash ADC modules in one of two forms, either threshold crossing (hit bits) or energy sum which is determined by the detector type. Fiber optic cables link the CTP to the Sub-system processors (SSP) which align and combine these data by detector system and forward to the GTP for trigger calculation. Not shown is the Trigger Supervisor (TS) which makes the final decision among the multiple trigger signals. It is linked to the GTP via a 32-bit parallel bus operating at the trigger clock rate of 250 Mhz.

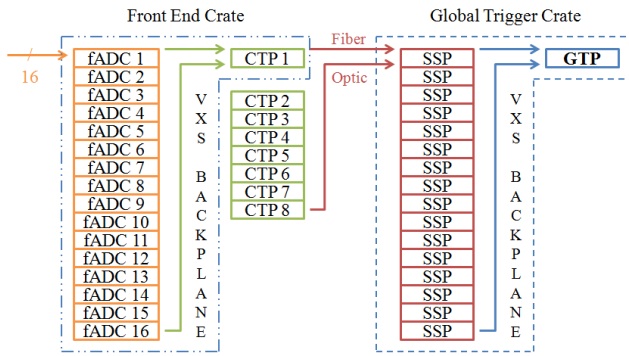


Figure 1: Level 1 Trigger System Data Flow.

User Interface

With an increase in complexity and quantity of trigger equations, the ability to intuitively configure them becomes more important. To help with the setup of trigger equations, a direct link to the GTP in the form of an Ethernet connection will be used. The interface will be a web page served from the GTP and will allow manipulation of coefficients and some adjustment to the form of the trigger equations. To provide greater levels of customization, reconfiguration of the FPGA is available using images in non-volatile memory. Users will be able to select between multiple pages already stored or upload new configurations.

VXS ELEMENTS

The VXS architecture has several advantages compared with other commercial off-the-shelf (COTS) chassis and backplane solutions. Making use of these elements has shortened the design process and reduced some of the associated risks.

VME Extension

Development on VME systems over the years has produced a set of verified hardware and software blocks which can be used in VXS platforms. Connectors P1 and P2 on VXS Payload modules remain unchanged from their VME counterparts allowing backward compatibility with existing designs. Designs consisting of VME and VXS modules can coexist in VXS chassis.

It is important to note that since the VXS Switch module was not previously defined in the VME standard, it contains no direct connections to VME boards, requiring at least one VXS Payload with P0 to enable communication between all modules.

VXS Architecture

VXS Payload modules include a high speed P0 connector in addition to the standard VME P1 and P2. This connector provides eight high speed differential signal paths and two single ended signals to each of the two VXS Switch modules specified by the VITA 41.0 VXS standard [2]. Several VXS Standards, including VITA 41.2, optionally refine these signals for specific communication standards as four transmit pairs and four receive pairs which are reversed on the backplane and two I^2C single ended lines. If full duplex communication is required, this convention is convenient to help reduce pin mapping errors [3].

VXS has a dual star architecture specifying two Switch modules, each with high speed and single ended connections to up to 18 payload ports [2]. Figure 2 shows the connections between the two Switch cards and a Payload. Where applicable, the transmitter and receiver are specified with respect to Switch A. In addition, transmit, receive, and I^2C designations are optional and specified in VITA 41.1 and 41.2 and are made to illustrate a possible configuration [4].

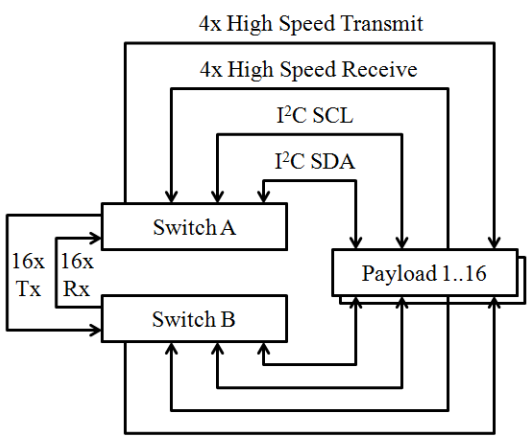


Figure 2: Example VXS Signal Mapping.

HARDWARE IMPLEMENTATION

Ethernet

The inclusion of Ethernet in an embedded system requires several hardware and software components to implement. Full TCP/IP over Ethernet is normally processed by a microprocessor running an embedded operating system and an Ethernet stack, a collection of functions related to the processing of Ethernet packets. Depending on the level of Ethernet implementation required, a simpler protocol such as UDP (user datagram protocol) can be implemented, saving costs on development tools and licenses. Non-volatile memory is required to store microprocessor boot code while fast on-chip and DDR2 memory is needed for many functions including instruction and data caching as well as dynamic memory allocation, program code space, and packet buffering. The electrical interface includes an Ethernet jack, magnetics, and a physical interface device (PHY). The Ethernet MAC (medium access controller) is available as an FPGA IP core and can be used with both TCP/IP and UDP protocols.

Given the flexibility that must be built into the design at its early stages, the GTP will use its FPGA resources, either Altera's embedded soft processor Nios II or logic elements, to provide Ethernet capabilities. A processor design helps deal with uncertainty inherent to TCP/IP networks including packet loss, retransmission and packet reordering. A pure logic element implementation of UDP benefits from very high throughput and efficiency but may prove impractical if the network quality impacts the data integrity.

VXS also allows for Gigabit Ethernet over the backplane using high speed signals. Due to the rates involved and FPGA capabilities, only simple circuitry is required. This consists of LVDS transceivers to buffer the backplane connection from the FPGA. A second processor may be instantiated in the FPGA to establish this link, minimizing the initial hardware investment.

External Memory

The GTP prototype contains both volatile and non-volatile memory in the form of DDR2 and NOR flash devices, respectively. A total of 512 Mb of flash storage and 2048 Mb of DDR2 memory are onboard.

The DDR2 memory is available in two 1024 Mb (128 MB) devices with fully independent address and data busses to accommodate two separate Ethernet interfaces, each with its own processor and memory buffers.

To hold multiple FPGA configurations, one flash device is logically segmented to contain several pages which are loaded using a CPLD (complex programmable logic device) configuration controller. The flash is programmed either via the Ethernet connection or through the front panel JTAG port. A second flash device is used to hold instructions and data for the CPU in addition to the Ethernet web server file structure.

Fiber Optic Transceivers

The CTP and GTP share many design aspects and could be used interchangeably when configured appropriately. To ensure full hardware compatibility, a four channel fiber optic transceiver was included on the GTP. These transceiver channels occupy four of the 36 lanes available on the FPGA. In cases where the additional functions of the GTP including Ethernet connectivity are required at the crate level, a GTP can be substituted. A firmware update would be required to provide CTP functionality in the FPGA.

Configuration

The CPLD's primary function is to provide safe configuration and reconfiguration control for the FPGA. It has access to a 256 Mb flash memory dedicated for storage of multiple FPGA images. The FPGA, which can also access the flash, can program images downloaded via the Ethernet interface and then command the CPLD to initiate reconfiguration. A safe FPGA image is stored without access by the user, allowing restoration of the GTP if the update over Ethernet fails.

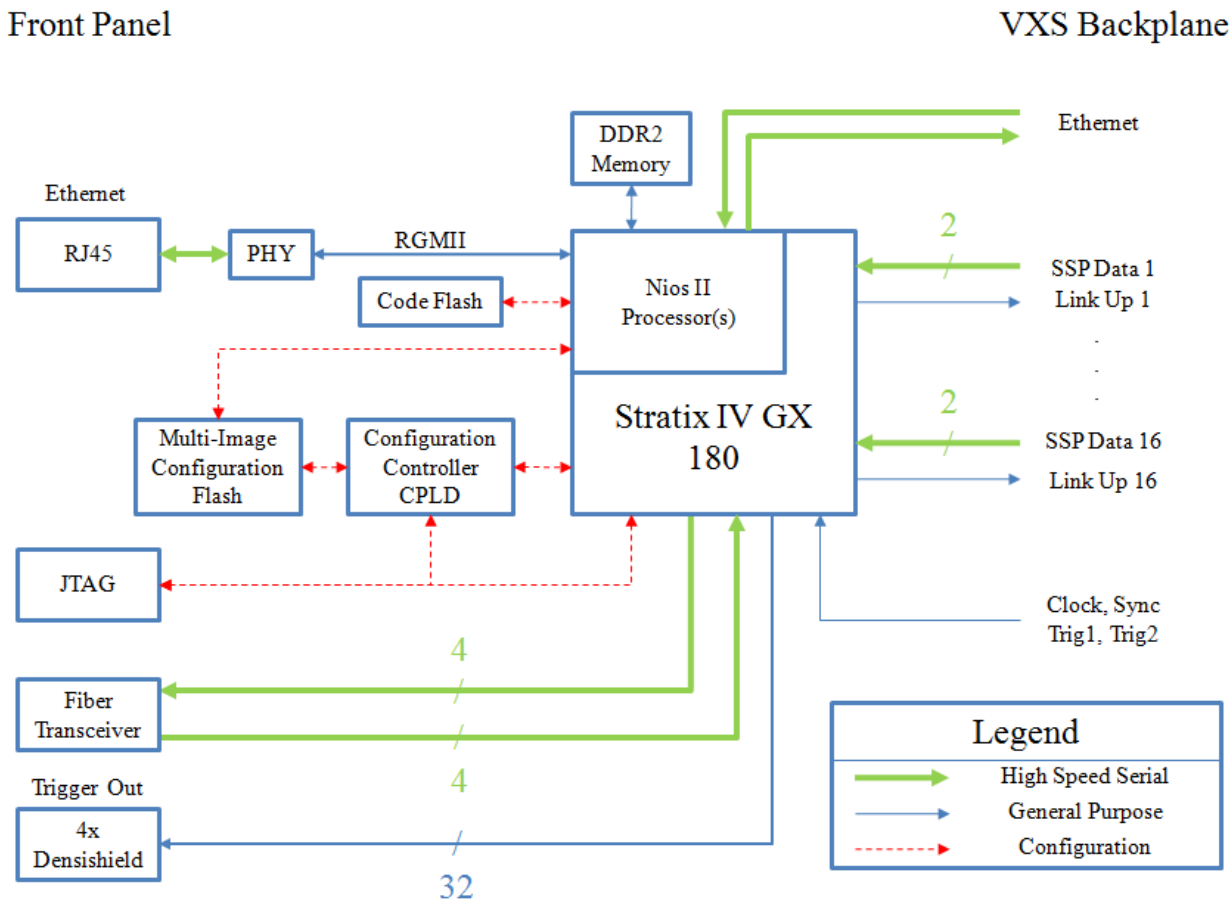


Figure 3: Simplified Global Trigger Processor Hardware Block Diagram.

Transceivers

In order to support the bandwidth and quantity of high speed signals arriving from the 16 SSP modules, an FPGA with integrated transceivers is the perfect solution. Using discrete devices with large parallel busses would not be practical given the number of pins and traces that would be required.

The selection of the Altera Stratix IV GX was made based on several criteria. While rated for speeds greater than 6 Gbps on all speed grades, the clocking structure places some limitations on a subset of the transceiver channels when all are utilized. With 32 transceivers allocated for receiving data from SSPs, four channels remain for use with the fiber transceivers, using all 36 lanes available on the FPGA.

A modified version of the Xilinx Aurora protocol has been selected for communication between the SSP and GTP. It is an open standard designed to encapsulate high speed links [5]. Because of the synchronization of the trigger data rate with the transceiver link rate, some of the Aurora protocol overhead has been removed including flow control and error handling. Lane initialization, channel bonding, and 8B/10B encoding are maintained. Low bit error rates are being targeted to account for the lack of error handling.

PCB Construction

While only a single GTP is required for each of the four experimental halls, efforts were made to keep the PCB costs to a minimum. The design was primarily driven by the interconnect required to tie all the components together and signal integrity, largely for the high speed signals and fast interfaces such as DDR2.

The FPGA requires almost a dozen different power rails between unique voltages and isolated supplies, driving the layer count for power planes. No power nets were run on signal layers and all are routed as split planes, minimizing impedance by providing wide plane connections. Power and ground on adjacent layers also adds high frequency decoupling for the FPGA and other sensitive components.

To maximize signal integrity, all signal and power layers are referenced to ground to provide close proximity, low impedance return paths. The result is a 16 layer board with six signal layers, four power layers and six ground layers. While conservative, all layers have critical signals which could be affected by inadequate or unbalanced routing. For example, matched DDR2 signals routed on two different layers could have increased skew and reduced margins without also matching return path.

With a transceiver link rate requirement of 5 Gbps, additional steps were taken to reduce stubs and minimize reflections. Since the concentration of trigger data requires data flow in only one direction, receive channels were given routing precedence. On incoming lanes from the SSP, all via stubs were eliminated by routing them on external layers rather than adding the cost of backdrilling. Only two full height vias were used, one for the VXS connector pin to the bottom layer and another from the bottom layer to the FPGA pad on top.

The VXS switch connector mapping distributes signals evenly by Payload Port across its four differential connectors. Since only two channels from each payload were used, this allowed all receive signals to be broken out on a single layer and all transmit lines on another.

Current Status

The GTP prototype is complete, shown in Figure 4, and hardware evaluation has begun. FPGA vendors provide many tools to help test hardware, in some cases with minimal development effort. In particular, Altera's Quartus II software comes equipped with a Transceiver Toolkit which allows rapid testing of transceiver channels. This allows manipulation of settings and data patterns in addition to bit-error rate (BER) monitoring and eye opening measurement to optimize the link signal integrity. Using a pseudo-random bit sequence (PRBS), the SSP to GTP links are performing well at 5 Gbps with PRBS31, a rigorous test pattern containing long strings of consecutive ones and zeros.

Altera's DDR2 memory controller core and External Memory Interface Toolkit help verify external memory data, address and command signals. Using a JTAG link between the PC and FPGA, calibration and margining can performed and the results reported. At a memory speed of 666 MHz, both memory devices pass by wide margins on all data and strobe lines.

Ethernet hardware development and TCP/IP software development are complex compared with most other interfaces. In order to provide the most flexibility, the GTP was designed to integrate all protocol layers above the physical interface within the FPGA. Altera's Qsys system builder allows the creation of customized systems by placing components in FPGA logic using a graphical user interface (GUI) and defining the interconnect. The architecture which is normally contained in a special purpose IC must now be developed and debugged in order to begin the Ethernet hardware evaluation.

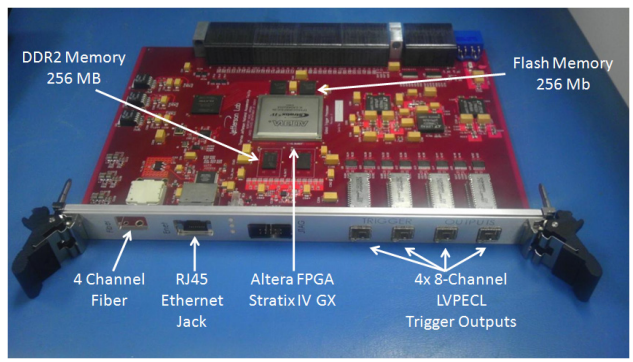


Figure 4: GTP Prototype.

CONCLUSION

Jefferson Lab's new Global Trigger Processor is a powerful, flexible platform well suited to handle the processing and communication requirements of the upgraded trigger system. It leverages the VXS architecture to build on both the experience and hardware of legacy VME systems. However, the GTP is also capable of a myriad of other tasks given its interconnect with the payload boards within the VXS crate, Ethernet and fiber optic connections to external devices, and plentiful onboard storage. While still in development, the GTP promises to adapt to evolving demands throughout the duration of the 12 GeV experimental program.

REFERENCES

[1] C. Cuevas, B. Raydo, and S. Kaneta, "Description and Requirements for the VXS Global Trigger Processor," D00000-16-08-S005, Jefferson Lab (2011).

[2] "VXS – VMEbus Switched Serial Standard," ANSI/VITA 41.0 (2006); http://www.vita.com

[3] "VXS 4X Serial RapidIO™ Protocol Layer Standard," ANSI/VITA 41.2 (2006); http://www.vita.com

[4] "VXS Overview," VMEbus International Trade Association, (2010); http://www.vita.com/home/MarketingAlliances/vxs/VXS%20Overview.pdf

[5] "Aurora 8B/10B Protocol Specification," SP002, v2.2, Xilinx (2010).

MEASURING ANGLE WITH PICO METER RESOLUTION

P. Mutti*, M. Jentschel. T. Mary, F. Rey, Institut Laue-Langevin, Grenoble, France,
G. Mana, E. Massa, INRIM, Turin, Italy

Abstract

The kilogram is the only remaining fundamental unit within the SI system that is defined in terms of a material artefact, more specifically a PtIr cylinder kept in Paris. Therefore, one of the major tasks of modern metrology is the redefinition of the kilogram on the basis of a natural quantity or of a fundamental constant. The Institut Laue-Langevin (ILL) and its high-resolution γ-ray spectrometer GAMS with its new optical interferometer can play a crucial role in this attempt.

INTRODUCTION

In the effort of redefining the kilogram one must approach a 10^{-8} relative accuracy in its practical realization. A joint research project amongst the major metrology institutes in Europe has proposed the redefinition of the kilogram based on the mass of the ^{12}C atom. The goal can be achieved by counting in a first step the number of atoms in a macroscopic weighable object and, in a second step, by weighing the atom by means of measuring its Compton frequency ν_C. It is in the second step of the procedure, where the ILL is playing a fundamental role with GAMS, the high-resolution γ-ray spectrometer.

Energies of the γ-rays emitted in the decay of the capture state to the ground state of a daughter nucleus after a neutron capture reaction can be measured with extremely high precision via Laue-diffraction. In order to match the high demand in angle measurement accuracy, a new optical interferometer with 10 pico radian resolution, linearity over a total measurement range of 15° and high stability of about 0.1 nrad/hour has been developed. To drive the interferometer a PLL circuit has been implemented for the heterodyne frequency generation. As the same time a new FPGA based electronics for real time phase measurement and axis-positioning control has been realized.

In the present paper, the basic concepts of the FPGA implementation will be revised and a complete analysis of the performances of the new electronics will be presented.

PHYSICAL MOTIVATION

Two approaches are currently proposed in order to be able to link a new definition of the kilogram to a fundamental constant.

The first project, the so-called "Watt-balance" [1], suggests linking the kilogram to Planck constant h. The second approach, the so-called "crystal density" [2], proposes to measure Avogadro constant N_A and define the kilogram via the mass of a free carbon atom: $1000/12 \cdot N_A \cdot A_r$ (^{12}C). Additionally one could also determine the energy equivalent to the mass of a ^{12}C atom, leading to the fundamental constant $N_A \cdot h$.

A direct measurement of the energy equivalent is difficult since small masses correspond to enormous energies. However, measuring the mass and energy balance in a thermal neutron capture reaction offers an attractive possibility to realize this concept. The key difficulty in the experiment is the precise absolute determinations of γ-ray energies emitted after the thermal neutron capture reaction.

This is done by diffraction of γ-rays at perfect Si crystals and measuring the very small diffraction angles in absolute terms. The aimed uncertainty for the determination of $N_A \cdot h$ is 10^{-8}, which requires an adequate accuracy for the diffraction angles' measurement and the calibration of the apparatus.

A key step towards a new $N_A \cdot h$ measurement is a new optical interferometer with 10 prad precision and linearity over a total measurement range of 15° and high stability of about 0.1 nrad/hour. Figure 1 shows the layout of the interferometer with all the optical elements in place and the various beams' paths.

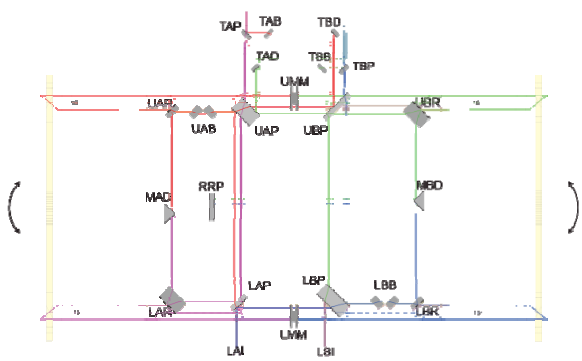


Figure 1: The optical layout of the new GAMS interferometer.

The interferometer follows a Mach-Zehnder layout, which allows a strict separation in space between all beams. This avoids unwanted mixing, which would lead to non-linearity in the angle measurement. A more detailed description of the interferometer can be found in Ref. [3].

The interferometer is designed such that we have the possibility to measure the rotation of both spectrometer axes using a single set of optical elements. This highly symmetric layout increases the stability and the accuracy of the angle measurement by self-compensating eventual drift of the apparatus. All the optical elements are high quality fused silica pieces with custom-made surface treatment. All the key optical elements of the

interferometer are chemically bonded [4] to the CLEARCERAM base of the spectrometer to avoid the use of glue. The roof-prisms and retro-reflectors used in the realization of the optical path are hollow since in solid elements the path length is not constant with respect to rotation [5].

THE NEW LASER SOURCE

Classical interferometers allow typical displacement measurements around 10^{-2} of the optical wavelength. This resolution can be improved by a factor of 100 by using in each interferometer arm a different wavelength.

This concept is realized in our new interferometer. Starting from a highly stabilized monodyne laser source, the single laser wavelength is split into two beams and each of them is frequency shifted. The two frequencies are obtained by mixing to a beat signal of 100 kHz with a carrier frequency of 40 MHz.

We have developed a Phase-Locked Loop (PLL) circuit that allows mixing the reference signal of a commercial lock-in amplifier with a signal of a precision signal generator. The two frequencies are transferred to the laser beam via Acousto-Optic Modulators (AOM). After frequency shifting the laser beams are injected into the interferometer via glass fibres. The interference signal will be detected as a beat node modulation. The phase of this beat signal is proportional to the length variation of the interferometer paths originated by the rotation of the axis.

The phase gets measured relative to a reference signal generated within the interferometer thanks to a newly developed phase measurement electronics described in more details in the following chapter.

PHASE READ-OUT AND NANO POSITIONG ELECTRONICS

The interferometer delivers for each of the two axes a measurement signal and a reference signal. If one axis is rotated, the phase of the beating of the according signal changes with respect to its reference. In order to determine the absolute displacement of the axis in angular term one has to measure the phase change and count integer periods (where one period is equivalent to one optical fringe or to an angle of about 0.5 μrad). To perform those tasks we have developed a dedicated electronic module based on a VIRTEX-4 FPGA. For compatibility reasons with the existing ILL's electronics and for a maximum of flexibility, the implementation adopts the M-Module (ANSI/VITA 12-1996) [6] mezzanine standard (see Fig. 2). While the back of the mezzanine card contains the FPGA chip and the carrier I/F connector the front part implement all the analogue circuits necessary to treat all signals from the interferometer as well as all the front-panel connectivity.

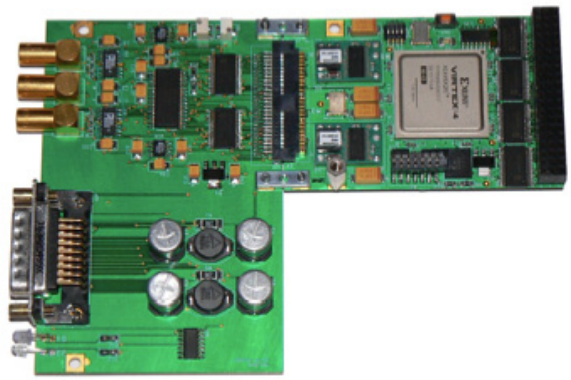


Figure 2: The M-Module responsible for the phase read-out and the fine positioning of the spectrometer's axis.

The beating signals from the photodiodes are first filtered to eliminate the DC offset and the narrow band AC noise due to the detection of ambient light. The signals are then transformed into square waves by high stability differential operational amplifiers. The phase difference between the reference and the interferometer signal is then measured by time interval counting. The module is driven by a master clock running at 100 MHz, which allows the 100 kHz periods to be sampled with an accuracy of ± 10 nsec. This uncertainty is equivalent to 0.0005 interferometer fringes (50 prad). The counting of integer fringes (equivalent to full periods) provides the necessary information for the macroscopic positioning of the spectrometer axes using DC motors coupled to high precision rotation stages. The accuracy of the phase measurement can be pushed up to 0.00005 fringes by means of a sliding average over 100 values, at the price of reducing the sampling frequency down to 1 kHz.

The axis fine positioning is achieved using a piezo actuator controlled by a 0-10 *V* signal provided by a 14-bit Digital-to-Analogue Converter (DAC). Alternatively, the card offers the possibility of a digital output to directly control the piezo positioning. At present a positioning resolution of 0.01 fringes has been obtained, being limited only by the amplitude of the residual external vibrations transmitted to the axis. To remove any axis drift and to minimize those residual vibrations, the phase information is used as feedback for a close loop real-time regulation mechanism. This regulation is based on hardware Proportional-Integral-Derivative (PID) controller that corrects for any deviation of the measured phase from the desired one.

The phase information is also recorded in coincidence with the γ-events from the HPGe detector in the acquisition card. This supplementary information is fundamental to correctly interpret the measured data since it allows removing the impact of mechanical vibrations that affects the accuracy of the angle measurements due to the incorrect determination of the axis position during the data acquisition.

CARD PERFORMANCES

A series of tests have been carried out to evaluate the overall performances of the ILL's phase read-out electronics in comparison with those provided by the commercial dual phase, wide bandwidth, DSP lock-in amplifier SIGNAL RECOVERY model 7280. The lock-in amplifier uses a different approach to derive the phase information. It uses frequency mixing in combination with a low-pass filter to convert the signal's phase and amplitude to a DC voltage signal. Its main advantage is the capability to handle signals with a high signal-to-noise level.

All tests for which we report the results in the following were performed using a high precision signal generator feeding both the reference and the interferometer signal therefore, the expected phase difference should be equal to zero. Figure 3 shows the evolution of the phase measurement when the beating frequency is increased from 1 kHz up to 120 kHz.

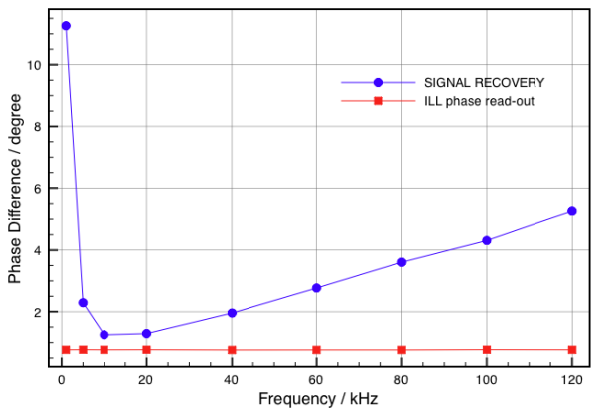


Figure 3: Comparison of the phase offset versus beating frequency.

Having in mind that both inputs are fed with the same beating frequency the expected phase offset should remain constant. With the exception of the first 2 points, the results show clearly a linear correlation between the frequency variation and the phase offset values in the case of the SIGNAL RECOVERY lock-in amplifier. Our explanation is that the observed behaviour is due to the filters and amplifiers that are present at the input stage. For very low frequency, it is evident in the data how the two input channels of the SIGNAL RECOVERY seem to diverge. We do not have a clear explanation for such behaviour. The same measurement repeated using our read-out electronics does not show any appreciable variation of the measured phase as a function of the beating frequency.

A second series of tests has been performed to verify the stability of the phase values with respect to amplitude variations of the input signals. The obtained results are summarized in Fig. 4 where the phase difference has been plotted versus input signal amplitude. As for the previous measurement, the same input signal has been used for both inputs resulting in an expected zero phase difference. While in the case of our read-out electronics this result is confirmed a part from an initial offset, the SIGNAL RECOVERY lock-in shows a strong dependency between phase value and input signal amplitude.

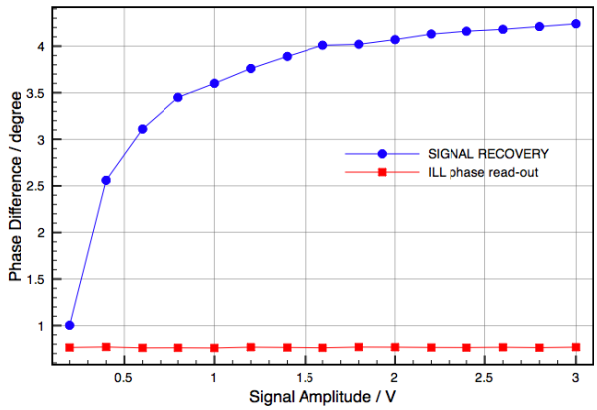


Figure 4: Comparison of the phase offset versus input signal amplitude.

In Fig. 5 and Fig. 6 we compare the power density spectra of the phase difference measured by both the SIGNAL RECOVERY lock-in amplifier and our electronics. The frequency resolution of the SIGNAL RECOVERY spectrum is 0.2143 Hz. The standard deviation of the SIGNAL RECOVERY phase noise (the *rms* value of the signal) - calculated by the square root of the integral spectrum in the 0.1 Hz to 500 Hz frequency interval - is 0.016 degree. The cut-off at about 100 Hz is due to the 1 ms minimum time-constant of the output low-pass filter (6 dB/octave).

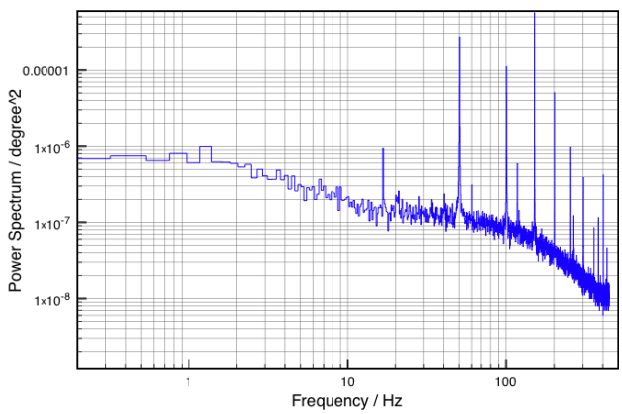


Figure 5: Noise power spectrum obtained with the SIGNAL RECOVERY lock-in amplifier.

The *rms* values are comparable for both the SIGNAL RECOVERY lock-in amplifier and our phase read-out electronics but since for this last one there is no cut-off at 100 Hz, the resulting bandwidth is larger and consequently the total noise lower. At lower frequencies it is evident, especially in the case of the SIGNAL RECOVERY, a slight noise level increase. The fact that the same effect is much less pronounced in our electronics

seems to confirm the previous results showing a much higher sensitivity to frequency and amplitude noise in the case of the SIGNAL RECOVERY.

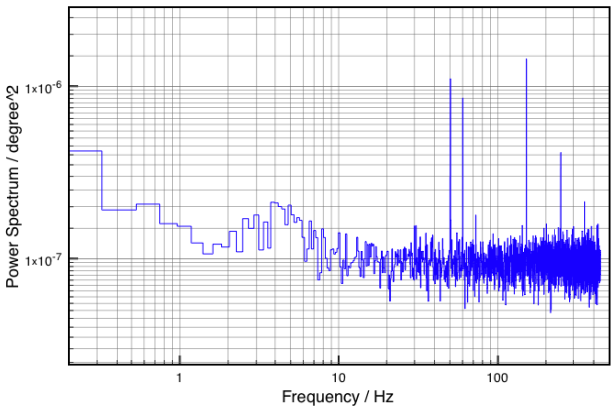


Figure 6: Noise power spectrum obtained with our phase read-out electronics.

From the SIGNAL RECOVERY data we can calculate an the *rms* noise level in the 0.1 Hz to 10 Hz interval of 0.041 degree. This value corresponds to 1.4 mV amplitude noise (*rms* values) of the signal generator in the same frequency band. The remaining noise present in our electronics is probably due the jitter of the quartz (50 ppm accuracy) used to cadence the FPGA and, therefore, to determine the time intervals for the phase measurement. To verify this hypothesis, a new version of the same card implementing a high precision quartz (30 ppb accuracy) is currently under development. Having established the good performance of our electronics, we continued the tests replacing the signal generator inputs with those obtained by a simplified version of our interferometer. The goal was to determine all possible sources of phase noise due to the laser injection and/or the optical fibre as well as to monitor the time stability of those elements.

CONCLUSION

In order to satisfy the specific requirement in terms of high precision phase measurement and positioning accuracy related to the operation of the new ultra high-resolution γ-ray interferometer under development at the ILL, we have specifically developed a new phase read-out electronics. The card adopts the M-Module mezzanine standard for a maximum of flexibility. It is based on a Virtex 4 FPGA that controls both the phase measurement and the fine axis positioning.

An exhaustive series of tests has been performed to evaluate the overall performance of the new electronics. The obtained results are very encouraging, showing a very small, when not absent, sensitivity to frequency and signal amplitude noise. A further step in noise reduction will be achieved with the implementation of the new high-precision quartz.

REFERENCES

[1] L.R. Steiner et al., Metrologia **42**, 431 (2005).
[2] P. Becker et al., Metrologia **40**, 271 (2003).
[3] J. Krempel, “A New Spectrometer to Measure The Molar Planck Constant”, PhD Thesis, Munich (2010).
[4] US-Patent 6,284,085 B1 “Ultra Precision and Reliable Bonding Method”.
[5] E. R. Peck, J. Opt. Soc. Am. Vol 38 No 12, 1015 (1948).
[6] ANSI/VITA 12-1996, "American National Standard for The Mezzanine Concept M-Module Specification", VMEbus International Trade Association.

NSLS-II BOOSTER POWER SUPPLIES CONTROL

P. Cheblakov, S. Karnaev, S. Serednyakov, BINP, Novosibirsk, Russia
W. Louie, Y. Tian, BNL, Upton, NY 11973, USA

Abstract

A special set of devices was developed at BNL for the NSLS-II magnetic system Power Supplies (PSs) control [1]: Power Supply Controller (PSC) and Power Supply Interface (PSI). PSI is placed close to current regulators and is connected to the PSC via fiber-optic 50 Mbps data link. PSC communicates with EPICS IOC (Input/Output Controller) through a 100 Mbps Ethernet port. The main function of IOC includes ramp curve upload, ADC waveforms data download, and various process variables control. The 256 Mb DDR2 memory on PSC provides storage for up to 16 ramping tables for both DAC channels, and a 20-second waveform record for all ADC channels. The 100 Mbps Ethernet port enables real time display for 4 ADC waveforms.

The NSLS-II booster PSs are divided into two groups: ramping PSs, which provide passage of the beam during the beam ramp in the booster from 200 MeV up to 3 GeV in 300 ms time interval, and pulsed PSs, which provide beam injection from the linac and extraction to the Storage Ring.

This paper describes a project of the NSLS-II booster PSs control. Characteristic features for the ramping magnets control and pulsed magnets control in a double-injection mode of operation are considered in the paper.

INTRODUCTION

NSLS-II 3rd generation light source [2] is currently being constructed at Brookhaven National Laboratory, USA. It will be a storage ring operating at 3 GeV. To provide an effective continuous operation of the storage ring a full-energy booster [3] is proposed. The booster will accelerate electron beam from 200 MeV to the nominal energy with average beam current of 20 mA. The booster will be capable of multi-bunch and single bunch operation. Maximal number of bunches in one accelerated train is 150.

Two modes of the booster operation are planned: 1 Hz cycle with two beam injections with a 100-ms interval and 2 Hz cycle with one beam injection. The energy ramp period is 300 ms in both modes.

The required relative accuracy of the main PSs (dipoles, quadrupoles) control is 10^{-4} for the current plateaus (at beam injection and extraction) and 10^{-3} for the ramp.

Power supply control system will provide beam passing in the booster during acceleration. It will synchronously control all magnetic elements for this purpose. Also it will provide operations on orbit correction at any moment of injection, acceleration and extraction, tune and chromaticity adjustment at any moment of beam existence in the booster, response matrixes measurements, and power supplies and magnet parameters monitoring.

Power supply control should be synchronous with other accelerator subsystems to perform its functions. The synchronization is done through timing system [4]. The PS control equipment receives a trigger pulse from the timing system and uses 10 kHz signal from Event Receivers, which is synchronous to the main RF oscillator. This synchro signal allows providing the required relative accuracy of PSs at the ramp.

BOOSTER POWER SUPPLIES OPERATION

Two modes of operation are supposed for the booster: a one-second cycle with double injection with a 100-ms interval, and a 0.5-second cycle with single injection. Beam ramp time is the same for both cases: 300 milliseconds. Typical current diagrams for different PSs are provided in Fig.1.

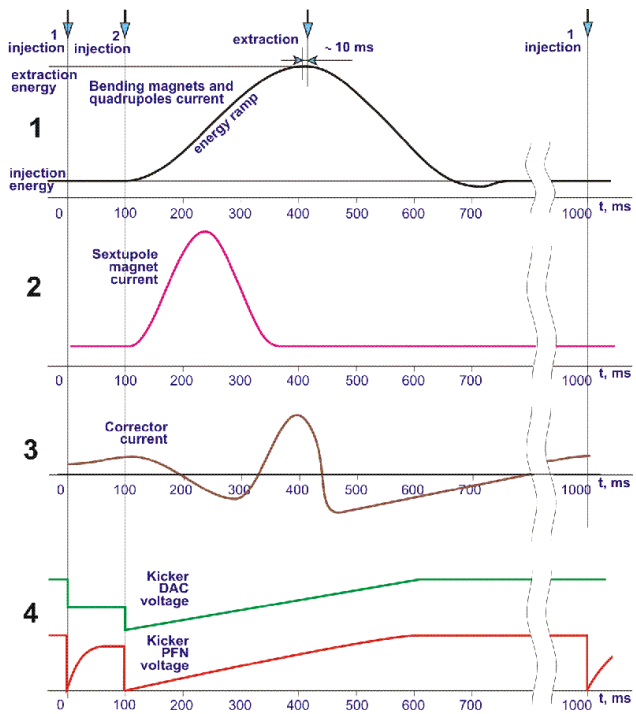


Figure 1: Booster elements current and voltage diagrams.

Diagram #1 shows change of bending magnet current in accordance with change of the beam energy. Depth on the plot in the region of 700 msec represents closure of magnetic loop for the dipole magnets.

Diagram #2 represents sextupole current compensating chromaticity deviation due to eddy current and magnet saturation effects.

Diagram #3 shows bipolar change of corrector current. The corrector ramp function should be changed in real

time at each booster cycle to provide an effective beam orbit correction.

Diagram #4 represents control and measuring signals for the injection kicker. Use of PSC allows change of DAC voltage during 100-ms interval synchronously with beam injections in hardware.

BOOSTER POWER SUPPLIES CONTROL STRUCTURE

There are 58 ramping PSs: 3 PSs for 3 groups of dipole magnets, 3 PSs for 3 groups of quadrupole magnets, 16 sextupole PSs, and 36 PSs for corrector magnets. Beam injection and extraction are provided by 9 pulsed PSs (injection and extraction septums, extraction bump magnets, 4 injection and 2 extraction kickers) and one DC extraction septum PS.

44 sets of PSC-PSI are proposed to be used for PSs control: 6 dual-channel sets for bending magnets and quadrupoles, 28 dual-channel sets for sextupoles and correctors, 10 single-channel sets for injection/extraction magnets. All devices will be placed in racks in the injection service area near the booster tunnel. A block-diagram of the booster power supply control is presented in Fig.2.

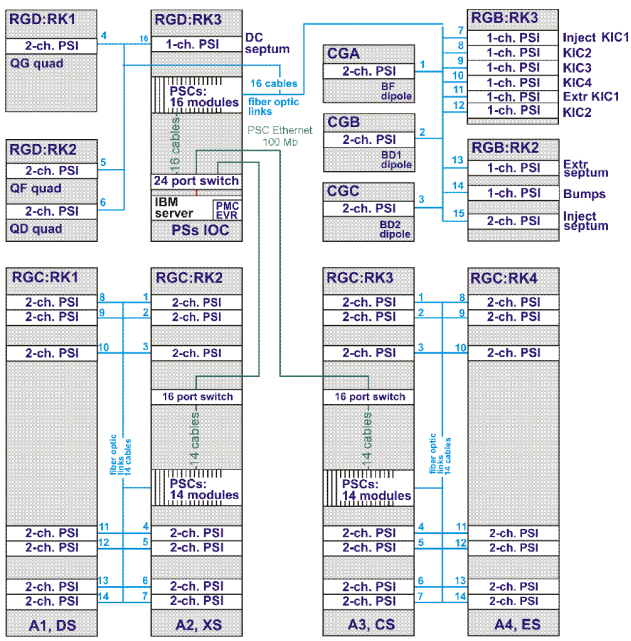


Figure 2: Block-diagram of the booster power supplies control. Distribution of the equipment among the racks.

It is supposed to use one IBM server (PSs IOC) to control all booster PSs. All PSCs will be distributed in three chassis and will be connected via switches in 1 Gbps network. .To maximize the communication speed, each PSC is directly connected to IOC via the Ethernet port. This will be the same for NSLS-II storage ring power supply control [1].

Ethernet interface of PSC is 100 Mbps, but it can only receive less than 13-14 kbyte ramping data per second due to the limit of FPGA softcore CPU, softcore MAC and a free light weight TCP/IP stack.

PSC-PSI DESCRIPTION

Main Parameters

PSI has one or two precision 20-bit DACs, three channels of precision 16-bit ADC for each DAC, six channels of 16-bit ADC for each DAC, sixteen digital inputs, 8 digital outputs. PSI is connected to PSC via fiber-optic 50 Mbps data link.

PSC is equipped with 256 Mb DDR2 memory, which provides sixteen 40 kb ramp tables for each ADC channel and a 20-second waveform record for all ADC channels. PSC has a 100 Mbps Ethernet port for connection with control system.

Operation Description

PSC version described in this paper is designed for the booster control. It has a short set of commands for interaction with high level applications:

- "Rst" resets all PCS registers to the initial state.
- "Enbl" is used for PSC disabling/enabling.
- "RampEnbl" is used for disabling/enabling of ramping mode.
- "BrkCmd#" breaks current ramp table processing and brings DAC# at "BrkRate#" (see below) rate to "BrkEnd#" state.

To provide a flexible operation PSC has the following set of setting parameters:

- "Mode" switches mode of PSC operation: Stop/Ramping Mode/Static Mode. Static mode is used for control of DC PS in the case when a transient process in PS due to a new DAC set is unimportant.
- "DoutMode" switches digital output mode: Static (level)/Pulse. Digital output pulse mode allows setting digital output to "High" level for specified time, then the level will be returned back to "Low" value.
- "DoutPlsWdt" - pulse width (ms) in digital pulse mode.
- "ClkSrc" - updates clock source: external (from backplane)/internal 10 kHz.
- "ClkDivd" - clock divider factor (0 - /1; 1 - /2; …. 254 - /255) for changing of the ramp time.
- "ActvTbl#" selects DAC# active ramping table (0-15 for DAC1, 16-31 for DAC2).
- "InActvTbl#" selects inactive ramping table for download.
- "BrkRate#" sets DAC# break rate.
- "BrkEnd#" sets DAC# break end setpoint.
- "AdcID#" selects ADC channel for #-waveform reading.

Also a set of indication parameters can be read from PSC:

- "PscLoadingRamp" - PSC ramping table download indication (Loading or Done).

- "PscFrountT" - PSC main board temperature.
- "PscRearT" - PSC rear board temperature.
- "PscIP" - last byte of PSC IP address (192.168.1.x).
- "PscDDR2Check" - DDR2 memory check during bootloading (Good/Failed).
- "PscFiberSd" - status of fiber link between PSC/PSI (Good/Failed).

PSC firmware provides reading/writing of the following values:

- Writing eight values of digital outputs and two DAC setpoints (DAC setpoints are used for static mode of operation).
- Uploading from IOC selected (determined by "InActvTbl#") ramping table of 40 kbyte data for ramping mode of operation.
- Reading sixteen digital inputs and eighteen ADC values.
- Downloading to IOC of two 10 kHz ADC waveforms selected from ADC1-ADC16 in real time.
- Downloading to IOC of two 1 kHz ADC waveforms for both DACs in real time. The order of ADC channels in the waveforms: ADC1,2,3,4,5,6,7,8,17 for DAC1 and ADC9,10,11,12,13,14,15,16,18 for DAC2. Each ADC has 1280 points in the waveform.

PSI is capable of detecting the status change sequence of digital inputs with a 10 ns resolution.

The ramping mode of operation will be used for all booster PSs: both for ramping PSs, and for pulsed/DC PSs. In the case of DC PS (for example, DC septum) the ramp table will be filled by one setting value.

In each cycle each PSC starts processing of ramping function downloaded from IOC by common trigger from the timing system. Also each PSC receives 10 kHz clocks from EVR in order to provide mutual synchronization.

SOFTWARE

Software interface for interaction with PSC (see Fig. 3) is based on EPICS.

Figure 3: EDM screen for testing of PSC/PSI.

At the moment an EPICS driver for PSC is developed and all process variables are implemented, EDM screen is prepared for testing of all PSC/PSI signals and commands.

It is supposed to use this screen for the first tests of the booster PSs at the test stand in BINP. Additional scripts for uploading of ramp tables from text or MATLAB files will be used.

A Ramp Manager (RM) application for operation with PSs during commissioning of the booster is being developed now. RM will provide edition of ramp function tables, synchronization of different PSs tables, generation of ramp function tables and uploading it to IOC, storing/restoring tables, visualization of ramp functions and measured waveforms. An example of RM python screen is presented in Fig.4.

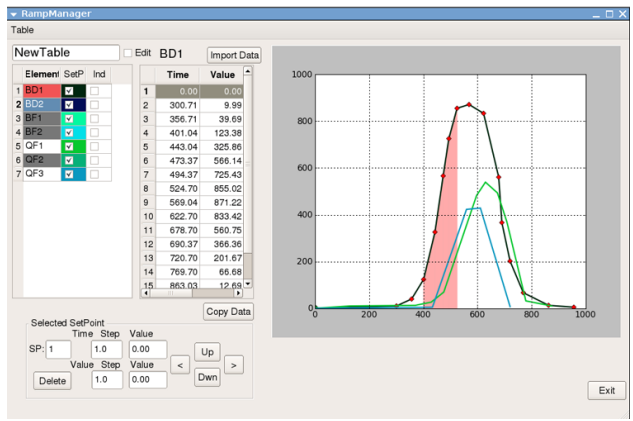


Figure 4: Ramp Manager screen.

SUMMARY

The detailed design of the NSLS-II booster PSs control is finished and all requirements are satisfied. Specifications for operation logic and parameters of Power Supply Controller and Power Supply Interfaces are worked out and are implemented in the firmware. Software is under development, its completion is planned before the start of the booster commissioning in October, 2012.

REFERENCES

[1] Y. Tian, et al., "Power Supply Control System of NSLS-II", Proceedings of ICALEPCS2009, Japan, http://accelconf.web.cern.ch/accelconf/icalepcs2009/papers/web005.pdf.

[2] Toshi Tanabe , "NSLS II Status Update ", IDMAX2010, Insertion Device Workshop, 9-10 November 2010, Lund, Sweden, http://www.maxlab.lu.se/usermeeting/2010/sessions/IDmax2010/IDMAX2010_pres/Session4/Tanabe.pdf.

[3] S. Gurov, et al., "Status of NSLS-II Booster", Proceedings of PAC20011, New York, USA, http://www.c-ad.bnl.gov/pac2011/proceedings/papers/wep201.pdf

[4] P. Cheblakov, et al., "NSLS-II Booster Timing System", paper at this conference, WEPMS015.

THE CONTROLLER DESIGN FOR KICKER MAGNET ADJUSTMENT MECHANISM IN SSRF

R. P.Wang, M. Gu, Z. H. Chen, R. Chen
Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China

Abstract

The kicker magnet adjustment mechanism controller in SSRF is to improve the efficiency of injection by changing the magnet real-time, especially in the top-up mode. The controller mainly consists of Programmable logic controller (PLC), stepper motor, reducer, worm and mechanism. PLC controls the stepper motors for adjusting the azimuth of the magnet, monitors and regulates the magnet with inclinometer sensor. It also monitors the interlock. In addition, the controller is provided with local and remote working mode. This paper mainly introduces related hardware and software designs for this device.

INTRODUCTION

Shanghai Synchrotron Radiation Facility, currently in routine operation stage for nearly two years, as a 3rd generation light source in China. SSRF storage ring will run in top-up operation mode[1]. Therefore, the SSRF injector will be in top-up injection .while in the Injecting, the electron beam which comes from the high energy beam transport line will be bumped to the closed orbit. To ensuring photon beam position stability, the beam in the storage must be remaining its orbit while travelling the injector. However, because of the manufacturing and installation differences, the magnetic field is not completely the same. This may make the beam deviate from the designed orbit, and reduce the injection efficiency. In order to reduce the impact of injection, we developed this device. It can be able to precisely measure the magnetic azimuth angle, and adjust it in real time.

ARCHITECTURE

Figure 1 shows the kicker magnet adjustment mechanism's architecture. In this system, the controller based on PLC, with the auxiliary hardware circuit, it selects the control object, adjusts the platform motion, detects the position and provides the protection. As the four kickers is not adjusted in the same time, we design one controller to adjust four platforms in instalments .By the switching, the controller can scan the current location of four the platform as well as focus on one. While in the scanning mode, the four position feedback information is received by the PLC cycle, and displayed on the control interface, and the power of motor is cut off. Otherwise, when in the adjusting mode, one of four kickers is focused, and the power supply of other kicker motor is cut off, so as to ensure the accuracy of selected objects, but also improve the safety performance. In addition, the stepper motors are equipped with brake system used as motion safety protection. Then there is an emergency, the motor is braking, and the platform is secured. Besides local interface, the controller also provides remote interface via Ethernet.

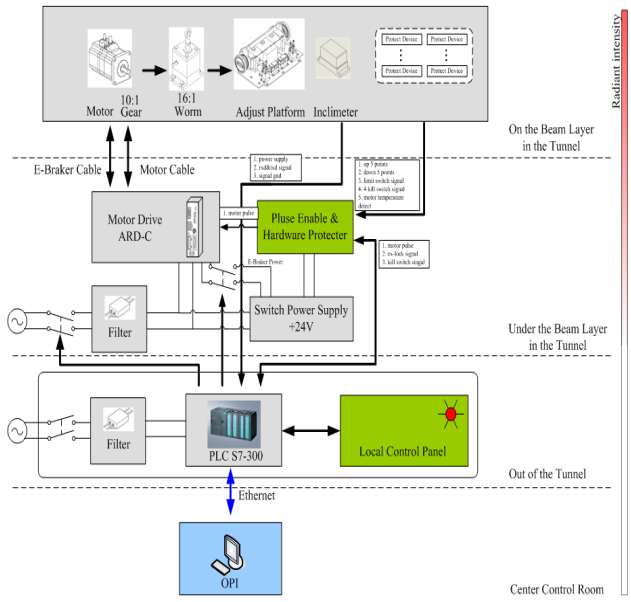


Figure 1: Adjust Platform Control System.

MECHANICAL STRUCTURE AND POSITIONING ACCURACY

Figure 2 is the motion scheme of the adjustment mechanism. The Platform is fixed on the point O. and the other underlay is the worm. When the stepper motor rotates, the worm controls points A movement up and down, then the magnet platform rotates slightly as the centre O, the relative position between magnet and beam then to be adjusted. In this Design, the motor step angle is 0.36°, and 1000 steps per revolution. With the mechanical design, the step motor can control the adjustment platform in less than 6 μrad。The speed of the motor is determined by the pulse frequency come from fm353. Considering the actual situation, the real adjusting range is very small, and the speed of the adjustment can be very slow.

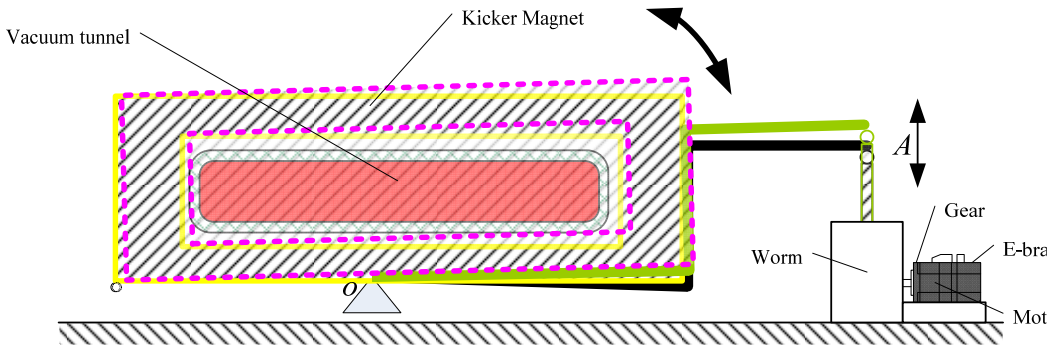


Figure 2: Motion scheme.

FEEDBACK CONTROL AND SOFTWARE DESIGN

Feedback Control

Position feedback is derived from inclinometer through optical fiber, through which PLC's SM340 module can acquire the real-time and high-precision position information. The benefit of inclinometers is the measurement is more direct with 2d angle and not subject to mechanical platform installation space conditions [2]. The inclinometer has many output forms such as RS232/RS484,CAN2.0,0.5-4.5V and 4-20mA. Furthermore, Inclinometer can measure the temperature and send it with the angle. With a view to actual position feedback form, special attention should be paid to check and confirm the increment direction of inclinometer during control system debug and assemble stage. If we set a wrong increment direction of inclinometer, we would get a decreasing angle but actually we need a increasing angle, and vice versa. Thus, such condition must be avoided.

The controller based on SIEMENS S7-300 PLC has so fast scanning speed and the adjusting range is so small that it can use close-loop control technology while the inclinometer is not very fast. During motors keep moving, PLC judges continuously whether motors have got to the destination position set by user through comparison between destination angle and current angle from inclinometer.

Safety Protection

With the exception of accurate motion control, the controller must ensure the safety protection. In this controller, software limit, kill switch and mechanical hard stop are used to implement up and down position protection. Software limits on the one hand prohibit setting value beyond the normal range; on the other hand, through comparison between software limits and feedback values from inclinometer, PLC must stop motor immediately and give warning indication. Kill switch can cut off pulses by hardware as well as give signal to PLC. Mechanical hard stops are used to stall the motor if the kill switch fails.

Software Design

Software designs are composed of two parts, one is program design based on PLC, and the other is program design based on EPICS.

PLC program design environments are SIEMENS Step 7 and WinCC Flexible 2008. Step 7 is used to develop motor control program and WinCC Flexible 2008 is used to develop application based on local panel. According to actual application, PLC's task modules mainly consist of motion control, inclinometer reading, safety protection, communication with EPICS and etc. Of all PLC's function modules, motion control is one key part. Because we utilize close-loop technology to realize accurate angle of magnet platform, we must adjust the motor speed into appropriate range in accordance with characteristic speed curve of stepper motor. Local panel program communicated with PLC via Ethernet mainly utilize WinCC Flexible 2008 to implement right status configuration and display.

Present EPICS environment is based on EPICS base-3.14.9 and Linux environment is Fedora Core 11. The communication between soft IOC and PLC is via Ethernet TCP/IP. To make EPICS save the last running statuses and parameters after IOC restarting, the autosave module is applied in the S7plc driver packages. Figure 3 is the interface in remote mode.

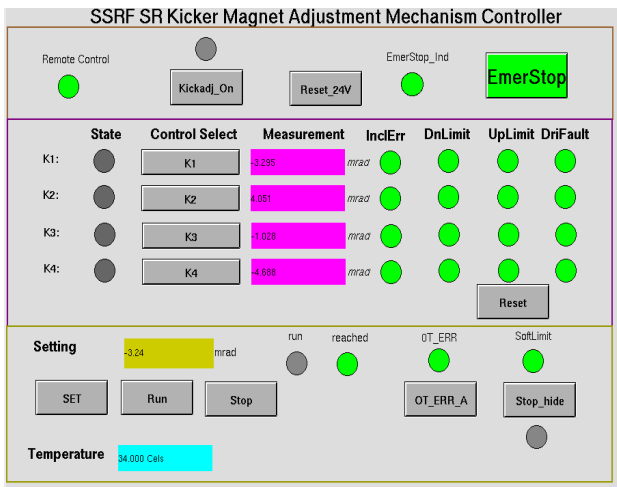


Figure 3: Interface in remote mode.

CONCLUSIONS

The adjustment mechanism has been installed on the storage ring injection kicker, and works well. The adjusting range can be achieved from -10mrad to -10mrad, the running accuracy is ±20urad. With the Software limit、kill switch and mechanical hard stop, the controller works in stable and reliable operation. By using this device to tilt the magnet, the rms value of the first 200 laps Track is less than ±4μm, which one can reached ±100μm before tilt optimized.

REFERENCES

[1] Z. T. Zhao and H. J. Xu, "SSRF: A 3.5 GEV SYNCHROTRON LIGHT SOURCE FOR CHINA", EPAC'04, Lucerne, July 2004, THZCH03, p. 2368 (2004); http://www.JACoW.org.

[2] SST500S inclinometers manual operation (2009).

ALBA TIMING SYSTEM - A KNOWN ARCHITECTURE WITH FAST INTERLOCK SYSTEM UPGRADE

Oscar Matilla, David Beltran [on leave], David Fernandez-Carreiras, Jerzy Jan Jamroz, Jorg Klora, Jairo Moldes, Ramón Suñé [on leave] ,CELLS-ALBA Synchrotron, Cerdanyola del Vallès, Spain

Abstract

Like most of the newest synchrotron facilities the ALBA Timing System works on event based architecture. Its main particularity is that integrated with the Timing system a Fast Interlock System has been implemented which allows for an automated and synchronous reaction time from any-to-any point of the machine faster than 5µs. The list of benefits of combining both systems is large: very high flexibility, reuse of the timing actuators, direct synchronous output in different points of the machine reacting to an interlock, implementation of the Fast Interlock with very low cost increase as the timing optic fiber network is reused or the possibility of combined diagnostic tools implementation for triggers and interlocks. To enhance this last point a global timestamp of 8ns accuracy that could be used both for triggers and interlocks has been implemented. The system has been designed, installed and extensively used during the Storage Ring commissioning with very good results.

INTRODUCTION

Alba [1] is a synchrotron light source under installation located nearby Barcelona. This 3 GeV third generation light source is planned to deliver the first X-rays beam to the users in 2012. The Linac has been commissioned in 2008, the booster in 2010 and the Storage Ring in 2011. The seven Beamlines included in the “Phase One” are being commissioned at the end of 2011.

The timing system is one of the critical systems in a synchrotron. Its main function is to synchronize the operation of the different parts of the accelerator. Including the injection procedure (Linac, Booster and Storage Ring) and the diagnostic devices.

When the Alba project started the initial triggering specifications were fixed by Accelerators Division. The main request was being able to generate triggers synchronously with the RF master frequency (499.654MHz in Alba case) in different points of the machine with a very low jitter (bellow 100ps), and to adjust the delay of some triggers in steps lower than 16ns.

Reviewing the state of the art of the timing systems developed in similar facilities it was found that two main approaches had been adopted historically: signal based model and event based model.

In the signal based architecture one reference frequency is downsampled to generate some trigger pulses that finally are distributed and delayed at each one of the destination points where it is needed to produce a trigger.

In the event based model the transmission is not of different pulses (or train of pulses) but of a code which is generated synchronously to a master frequency and transmitted following a typical start topology to all the points where the trigger output has to be produced. At each one of these points the continuous stream of these codes is decoded and a trigger pulse with an adjustable delay is generated if needed. One of the main difference between both approaches is that in the event based case all triggering information reach all the points of the facility. That involves full flexibility to change trigger parameters (as the decodification is remotely controlled) and a much easier maintenance. This is the main reason why all the newest synchrotrons have followed this architecture lately.

At some point other requirements were needed by Accelerators division about some trigger and fast interlock systems needs. The request could be divided to have two independent main characteristics: being able to produce synchronous outputs in different points of the machine and being able to react very fast to an interlock.

The idea of combining all those systems in a unique one which could give the maximum possible flexibility and performance came up. This paper explains how this system was defined as an upgrade of the Event based architecture and the process that was followed to implement and commission it.

STANDARD ARCHITECTURE

To establish a common framework a brief description of the different elements in the event based timing architecture will be done (please refer to Figure 1).

The first element is a device that generates the codified event stream (from now called EVG) synchronous to the RF frequency. In the case of Alba, it was also requested being able to generate events synchronous to a external 50Hz signal. The aim was being able to produce triggers synchronous to the AC 220V 50Hz power to minimize any possible effect in the ripple voltage of the power supplies for some specific triggers.

The event stream generated with the EVG is distributed following typical tree architecture to all the points where the trigger needs to be generated. To spread the signals there are some units called Fan Outs that produces multiple identical stream outputs from one event stream input.

Finally in the different destination points there are elements called Event Receivers (EVR from now) that decodes the stream. If some predefined event codes are received the trigger output with some preprogrammed characteristics (delay, pulse width, etc…) will be executed. The outputs will be synchronous to the event stream input and therefore to the master RF frequency in all the facility.

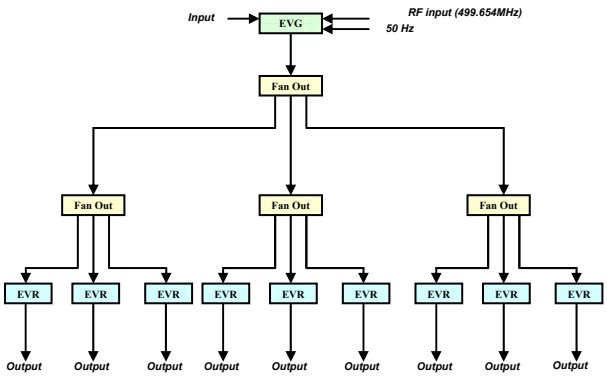


Figure 1:Event based timing architecture.

At Alba two different tenders were published: the first one included the electronics boards based in cPCI standard that were needed for each module (EVG, Fan Outs and EVRs). Among the bidders offers the contract was awarded to Micro-Research Finland Oy [2].

And in the second tender the optic fibers used for the timing network were included. More than 10Km of OM3 cable with 4 fibers were requested with a very low dispersion in the optic fibers length to avoid thermal effects in the delay change and also to minimize the delay adjustment procedure in the facility. The contract was awarded to Reichle & De-Massari (R&M)[3].

For very specific cases a delay adjustment within the bunch time was needed to be generated (in Alba case 2ns): Linac gun trigger, an streak camera used for diagnostic purposes and in the booster kickers magnets. For these cases a special product was requested to MRF which provided a special 6U cPCI card allowing delay adjustment of 10ps steps (EVRTG-300).

FAST INTERLOCK ARCHITECTURE

General Layout

When other triggering specification needs arose which could not be fulfilled with the Alba Equipment Protection System [4], as the capability to react synchronously to an interlock in different points in the machine or to react to an interlock in the sub miliseconds range, different approaches about how to fulfill such needs were considered.

The timing system already offered a platform to produce synchronous outputs and also the transmission of the triggers was fast enough. Moreover the timing network already covered more than 80 different points all over the accelerator. Therefore the idea of upgrading the timing system implementing a bidirectional communication link was considered as the best possible option.

Also there were different factors that reinforced following such direction: the first one was the high flexibility that this architecture would involve, the second was that the diagnostic tools of events timestamp that already were going to be implemented in the timing system could also be used for interlocks diagnostics. Doing in this way the timing diagnostic tools could become a very powerful tool to be used as it will be described later.

Finally from the cost point of view it appeared as a very good solution as the timing network was composed by four fiber cables (following a market "de facto" standard) which only one was used at that time.

Therefore the objective was being able to generate an independent event stream from the EVRs towards the EVG using the same timing protocol. That stream of events would be needed to be multiplexed in a new unit called “Fan Out Concentrator” which would become one of the main parts of this new architecture as it was needed to manage collision or saturation scenarios. Finally the interlock event stream would arrive to the EVG that would introduce the event in the normal timing event flow so the interlock could arrive to all the points of the machine following the same path as the rest of triggers generated by the EVG. Please refer to Figure 2 for a simplified architecture diagram.

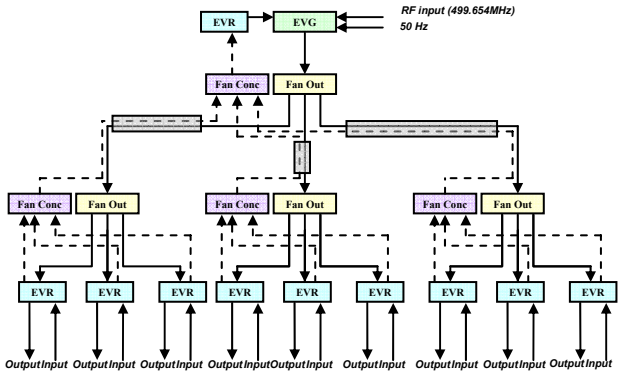


Figure 2: Fast Interlock System architecture.

To generate this system, MRF was contacted and a collaboration agreement was achieved that lead us to a new EVR firmware with all the capabilities implemented and with the development of a new unit by MRF: the Fan-Out Concentrator (cPCI-FCT-8).

Response Time

In a typical timing system the delay between the trigger generation at the EVG and its distribution to the EVR is not an important issue as far as all the triggers outputs are delivered synchronously. But if the timing system is required to react to interlocks it is needed to characterize the response time of the system.

In the hardware implementation we are using the codification/decodification process of the events uses 30 event clock cycles. In the case of Alba is 125MHz that is lead us a total is 240ns. So taking into account the number of stages present in Alba.

$$\text{time} \approx 240\ \text{ns} \times 7\ \text{stages} + 5\ {}^{\text{ns}}\!/_{\text{m}} \times 500\ \text{m} \sim 4.2\ \mu\text{s}$$

To understand the real magnitude of this number it has to be considered that the interlock propagation from any-to-any part to the machine and production of a reaction is a value closer to 4µs where 60% of the time is used just for signal propagation over the optic fiber. So no big margin for speed propagation improvement seems to exist for the future.

Timestamp

Each EVR has an internal time reference. In the case that a predefined event is received a timestamp log is possible to be acquired. This feature that is useful for triggering information it is still more interesting when interlock information can be acquired with common time reference being possible to match triggering effects and interlocks.

The same timing network is used to distribute the global clock reference to all EVRs. The system is based in a in a GPS receiver with a built in OCXO oscillator that provides PPS with an accuracy higher than ±100ns. Once this pulse is generated a reset for the seconds counter is sent via an event by the EVG. Everytime the reset is produced in a EVR an internal counter of 125MHz clock with accuracy higher than 40ppm is restarted.

For managing the PPS distribution and transmitting the UTC 32 bit seconds value an in-house electronic module was developed at Alba: the NTP Timestamp Server.

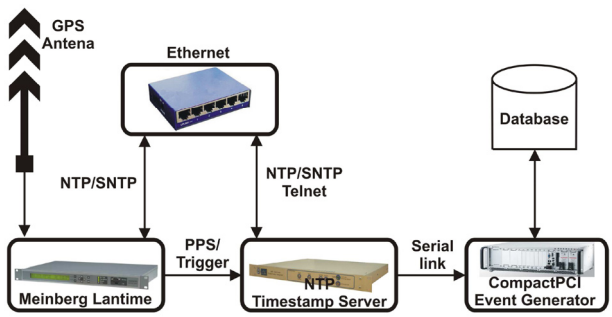


Figure 3: Global time reference distribution.

In the end at each EVR is available a timestamp of UTC time in seconds plus 8ns accuracy. So any event or interlock can be timestamped with 8ns accuracy.

APPLICATIONS

In this section a brief description of the practical applications where the Fast Interlock system has been used is presented.

RF Plants Fast Interlock

In total there are seven RF plants at Alba; six in the Storage Ring and one in the Booster. It has been developed in-house an electronic module that centralizes the safety conditions under which each one of the plants can operate (Figure 4). If those conditions are not fulfilled the plant is stopped locally and a different Fast Interlock event for each plant is generated. That event will execute in less than 5 µs all preprogrammed diagnostic and safety actions.

The unit can detect interlocks from three different RF detectors, 8 independent Arc detectors, one timing input (link to the Fast Interlock System) and a dry contact (normally used for slow signals as vacuum loss scenario detection). The reaction time is of 175ns with 5ns jitter and everything is fully programmable from its 10/100 Base-T Ethernet Port.

Internally an Alarm Log Engine has been implemented acquiring a maximum of 512 alarms sequences with 20ns timestamp that can be correlated with the global timestamp of the machine.

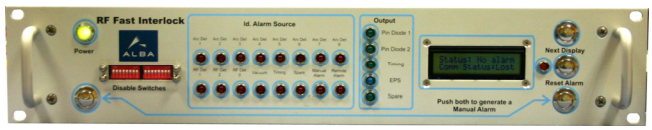


Figure 4: RF Fast Interlock Module.

BPM Fast Interlock

Alba has in total 176 BPM distributed around the 16 sectors of the Booster and the Storage Ring. Each one of the BPM is acquiring the X and Y orbit continuously. In case of an eventual distortion of the orbit outside certain predefined limits an interlock is generated.

In case that an BPM interlock is detected a Fast Interlock event is generated and will produce an immediate stop of the RF Plants and a distribution of a Post Mortem trigger to the rest of BPMs for acquiring a synchronous snapshot of the orbit. It has been defined 32 different fast interlock events: one for each one of the sectors of the SR and the booster. Doing in this way it can be easily identified the first sector where the orbit was disturbed which is a useful information for the Diagnostic group.

In the next figure a complete sequence of one BPM interlock timestamped with the Fast Interlock System is shown. A orbit interlock is detected in one BPM in SR in sector 13 and timestamped. The Fast Interlock system induces a shutdown of the six RF Plants 4µs later and that leads to a loss of the orbit in the rest of BPM sectors after 290µs.

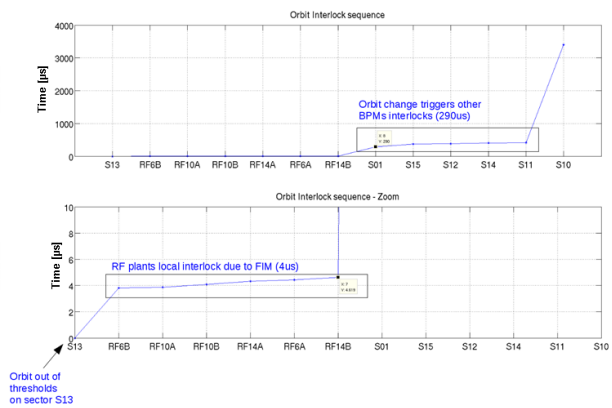


Figure 5: Orbit interlock sequence.

Front Ends

An independent Fast Interlock event has been defined for each Front End. In case of a vacuum loss scenario the RF plants will be automatically stopped to minimize the quantity of beam time that the vacuum gauges that closes had to absorb and to timestamp the beam loss.

CONCLUSION

In this paper a new architecture for the timing system has been proposed as an upgrade of the typical event based architecture that allows for fast interlock system implementation.

Several years have been needed since the initial idea came up until all the system has been developed, implemented, tested and commissioned. Currently the system has been working without problems for more than one year giving excellent service to Accelerator Division for the machine operation and completely integrated in the Control System (please refer to [5] for detailed information as it has been decided to detail the software implementation in an independent document).

Among the list of benefits one would highlight its very fast reaction time achieved that currently seems difficult to improve as 60% is spent in the transmission of the signal over optic fibers, the extremely high flexibility and reconfigurability, easy maintenance, its relatively low cost and the global combined timestamp tool that can be used both for triggers and interlocks using a common time reference.

CONTRIBUTION

Many people have contributed to this project over the time. We would like to thank to the whole Accelerator Division specially to F.Pérez, A. Olmos, U. Iriso and A. Salom[on leave]. Also to S. Astorga, X. Fariña, J. Pagés, A. Ruz and B.Saló for their dedication to the timing network check. To Y.Chernousko (Diamond Light Source) for his inestimable help in the first steps of this trip. And to J. Pietarinen from MRF [2] for getting involved in this project with the technical excellence we are already used from him.

REFERENCES

[1] CELLS-ALBA. http://www.cells.es
[2] Micro-Research Finland Oy. http://www.mrf.fi/
[3] Reichle & De-Massari http://www.rdm.com
[4] D. Fernandez-Carreiras et al., "Personnel Protection, Equipment Protection and Fast Interlock systems. Tree different technologies to provide protection at three different levels", ICALEPCS'11.
[5] J.Moldes et at., “The MRF timing System. The complete control software integration in Tango, ICALEPCS’11.

ALBA HIGH VOLTAGE SPLITTER - POWER DISTRIBUTION TO ION PUMPS

J. Jamroz, E. Al-dmour, D. Beltran, J. Klora,
R. Martin, O. Matilla, S. Rubio-Manrique

CELLS-ALBA Synchrotron, Cerdanyola del Valles, Spain

Abstract

High Voltage Splitter (HVS) is an equipment designed in Alba that allows a high voltage (HV) distribution (up to +7kV) from one ion pump controller up to eight ion pumps. Using it, the total number of high voltage power supplies needed in Alba's vacuum installation has decreased significantly. The current drawn by each splitter channel is measured independently inside a range from 10nA up to 10mA with 5% accuracy, those measurements are a base for vacuum pressure calculations. A relation, current-pressure depends mostly on the ion pump type, so different tools providing the full calibration flexibility have been implemented. Splitter settings, status and recorded data are accessible over a 10/100 Base-T Ethernet network, none the less a local (manual) control was implemented mostly for service purposes. The device supports also additional functions as a HV cable interlock, pressure interlock output cooperating with the facility's Equipment Protection System (EPS, ref: [1]), programmable pressure warnings/alarms and automatic calibration process based on an external current source. This paper describes the project, functionality, implementation, installation and operation as a part of the vacuum system at Alba.

INTRODUCTION

Alba's vacuum system contains 458 7kV ion pumps, 392 of them were selected to be supplied by HVS-s, it means that 392 HV outputs had to be distributed in the different positions of the machine, in the most efficient way and reducing costs for the HV cables. The best installation topology was designed and finally 83 HVS units were produced and installed, "20x8channel" supplying the booster and "63x5channels" supplying the storage ring, front ends and linac.

High Voltage Splitter was a collaborative project between Q-Lambda company [4] which developed a current measuring solution (sensor) under the high voltage, Vacuum Section in Engineering Division which provided necessary theoretical assumptions and calculations, R&D Section in Engineering Division which managed mechanical design, Electronics Section in Computing Division [2] which developed the project, managed a HV current sensor evaluation, upgrades and troubleshooting, and Control Section in Computing Division which developed a TANGO control system.

IMPLEMENTATION

HVS contains two systems: low voltage electronics which manages data processing, control, communication, and high voltage electronics which supplies ion pumps, splits the high voltage and measures the current consumption of ion pumps.

A typical HVS application is presented in Figure 1. The standard arrows show an Ethernet data flow, while a user records pressure measurements each second. The big arrows depict the high voltage distribution from one high voltage power supply to eight ion pumps. The checked arrows provide a basic implementation storing time related data into a database.

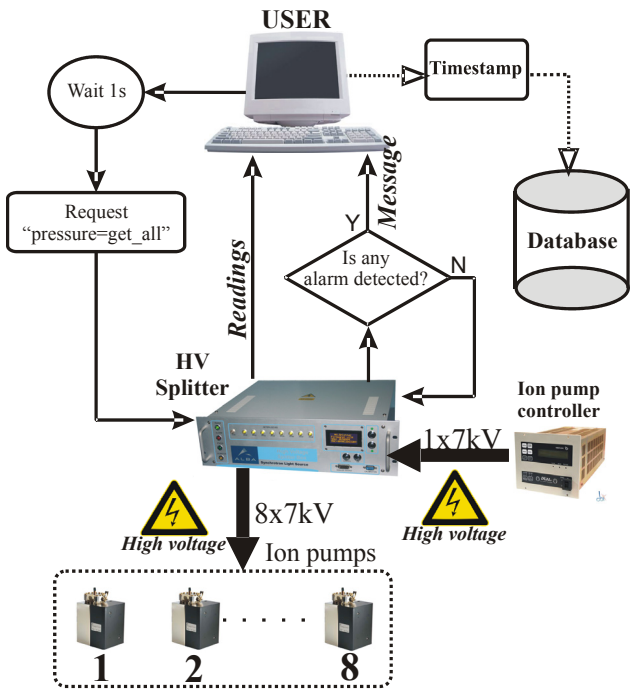


Figure 1: Typical application.

During this process, the current drawn by each HV channel is measured independently.

PRESSURE MEASUREMENT

Extensive performance evaluations under high voltage conditions have been done to proof the functionality. The pressure measurement involves a current consumption conversion into a pressure reading under constant conditions which are configured and predefined by a user,

and determined by a high voltage value and ion pump model.

Theory

A real correlation example current-pressure describing 25[lps] noble diode ion pump supplied with constant high voltage U=7kV is presented in Figure 2.

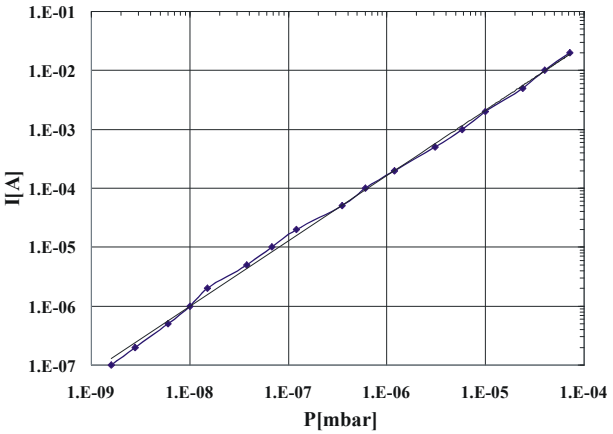


Figure 2: 25lps noble diode ion pump, Uconst=7kV.

The characteristics is almost linear and can be described by Equation 1. The chart, equation and related theoretical support were provided by Gamma Vacuum [3]- manufacturer of ion pumps. Generally all other pump models (25, 75, 150, 300, 500[lps]) strongly correlate with the theoretical formula.

$$P[\text{mbar}] = \frac{0.08778 * \frac{5600}{Voltage[\text{V}]} * Cal_{Factor} * I[\text{A}]}{IonPump_{Speed}[\text{l/s}]}$$

Equation 1: Default pressure implementation.

The HVS current measurements are based on Q-Lambda sensors [4]. The sensors are current-to-voltage converters which provide a linear voltage output (0-8V) related to a logarithmic current input (10nA-10mA). A real conversion is presented in Figure 3, the characteristic "V=f(I)" shows a difference in current readings acquired under low voltage using a current source reference and high voltage (7kV) supplying ion pumps.

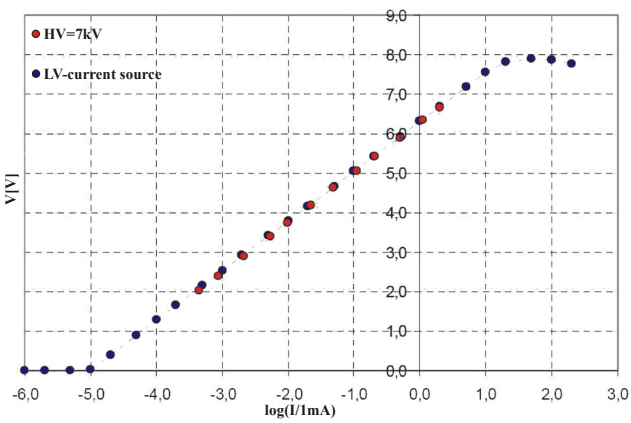


Figure 3: Current measurements "V=f(I)" under high and low voltage conditions with a Q-Lambda current sensor.

The output voltage dispersions are almost negligible, lower than 5%, so tests and calibration issues can be performed under low voltage without any quality risk.

Current Calibration Process

HVS integrates an automatic calibration process based on a Keithley 2611A source/meter working as a current generator/reference. The implementation is presented in Figure 4.

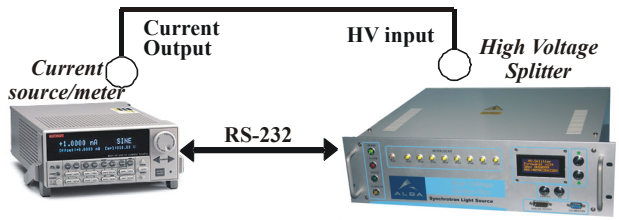


Figure 4: High Voltage Splitter calibration.

The Keithley current source is able to set a fixed current with an error lower than 0,001%. Two connections are necessary between those two instruments to execute the calibration: the serial RS-232 link and current link connected to a splitter channel under the calibration. Once the process is executed, HVS takes a control over the Keithley source sweeping all the current ranges and completing the process. For some specific ranges, HVS needs some time in order to compensate internal thermal drifts. This process is completely transparent for a user and assures 5% accuracy inside the range 10nA up to 10mA. The calibration data are fully available remotely for backup purposes.

Pressure Reading Implementation

A user can bind a pressure value to a current one making a set current-pressure, once the set is prepared, can be applied to any channel determining calculations of the final pressure readings (e.g. like in Figure 2, blue line). It means, a user can adjust HVS pressure readings with a full flexibility (even nonlinear characteristics), in total 13 sets can be programmed: one per each ion pump model (25, 75, 150, 300, 500[l/s]) and 8 independent ones. One set has to be tied to one channel, by doing it this way, a user can adjust the pressure readings of a channel for his particular real installation determined by real measurements, experience, or other gauge instruments, and not being fixed purely to the theory. If those sets has not been inserted, the default configuration is applied based on Equation 1 which gives very accurate readings as well.

ALARMS & PROTECTIONS

Pressure Alarms

Two different pressure alarms can be configured with two independent thresholds for each channel. If the

pressure reading of any channel exceeds the first threshold level, a warning message is visible in the splitter front panel and also the control system is informed with a proper warning event over Ethernet. The second pressure alarm activation provides the same warning/alarm messages, plus opens a dry contact output informing Alba's Equipment Protection System [1].

Equipment Protection System

An exceptional vacuum loss scenario is indicated with the dry contact EPS output. As this trigger shuts down the accelerator, a redundant activation mechanism has been implemented. It is possible to set up to 8 different EPS alarm configurations which contain alarm conditions for different ion pumps (channels). One particular configuration example is presented in Table 1, it shows that the machine will stop if the second pressure threshold is exceeded in the channels: Ch2,Ch3 and Ch8.

Table 1: EPS Alarm Configuration Example

	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8
EPS Config		X	X					X

HVS has also an external interlock input. Its purpose is to combine the EPS alarm with any other external interlock e.g. provided by any ion pump controller etc.

Cable Interlock

In order to increase security during the instrument operation, a cable interlock system has been implemented. An ion pump controller provides a cable interlock security mechanism which generates a 5V high impedance signal to interlock circuitry inside a high voltage cable. A functional diagram is depicted in Figure 5.

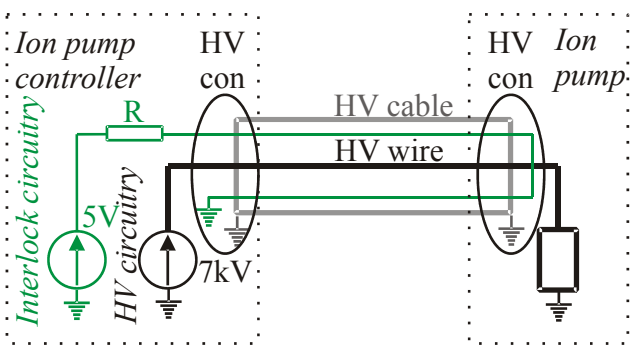


Figure 5: HV cable diagram for one ion pump controller and one ion pump.

When a high voltage cable is connected to an ion pump, a shortcut is applied to the interlock circuitry. In this way, the controller senses the high voltage cable status. If a low level is detected, it means that the high voltage cable has been correctly connected and the high voltage output can be enabled.

In case of HVS, a cable interlock distribution system has been designed to preserve this security protection. Internally all interlock signals are connected in series. It means that all high voltage cables connected to their splitter have to be correctly connected to ion pumps, otherwise the high voltage will not be enabled by a supply controller as the cable interlock protection will prevent that. HVS also detects which is a cable that is unconnected (failing) and reports it on its display and remotely. The HVS interlock distribution is depicted in Figure 6.

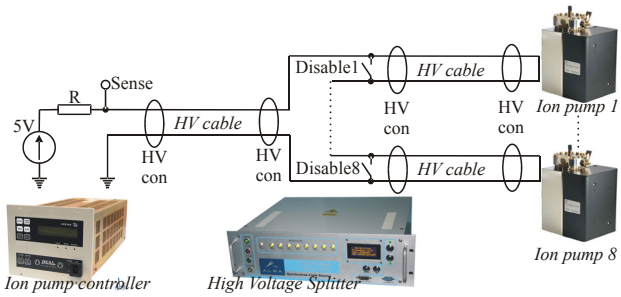


Figure 6: Cable Interlock diagram for one ion pump controller, HVS and 8 ion pumps.

CONTROL & OPERATION

Local Control

The local control is based on a keypad which allows to navigate between two submenus:

- "Measurement menu" - shows actual measurements.
- "Settings menu" - allows to change configurations.

Remote Control

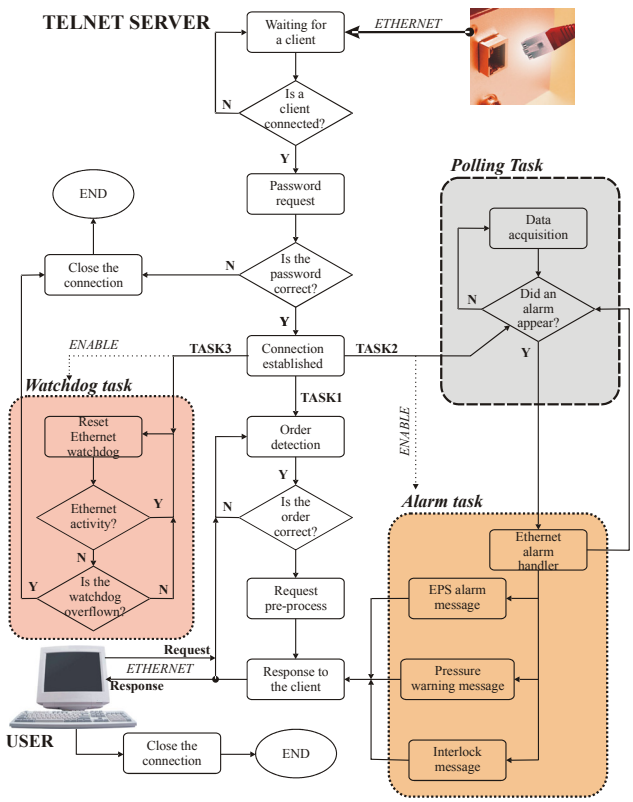


Figure 7: Telnet server – block diagram.

HVS provides its settings, status, measurements and records over Ethernet network. A DHCP protocol applies an automatic IP configuration and Telnet provides a bidirectional interactive text-oriented communication. Over 80 different string commands are available giving a full flexibility for any kind of implementations, plus an event based system informs a control room about any exceptional situations. The remote workflow is presented in Figure 7.

Operation Modes

HVS has two operation modes: commissioning and secure. In the commissioning mode, a user has unrestricted access to configuration settings using the local control menu. In the secure mode, the access to the configuration settings are limited only to Ethernet commands. A change from one mode to another can be done only remotely, so via the control room.

HVS is aimed to work in the secure mode. It assures that there is no accidental action done with the local control which could generate some alarms in the machine.

CONCLUSIONS

High Voltage Splitter has been a collaborative in-house project between many different sections: Vacuum, Mechanics, Electronics and Controls at Cells. The project development and tests took around one year, and first units were applied in first vacuum installation of the booster. From that point on, HVS-s have been fully working for more than 2 years also supplying the storage ring, front ends and linac. In total, 83 units were installed which made HVS a key instrument in Alba vacuum system.

Full implementation flexibility, complete integration to Alba's TANGO based control system, diagnostic purposes and cost efficiency make this project a great solution and improvement of synchrotron technologies.

REFERENCES

[1] D. Fernandez-Carreiras et al., "Personnel Protection, Equipment Protection and Fast Interlock systems. Tree different technologies to provide protection at three different levels", ICALEPCS'11.

[2] ALBA Computing & Controls Division, “Product catalog”, http://computing.cells.es/products/catalog.

[3] Gamma Vacuum, “High and Ultra High Vacuum Products”, http://www.gammavacuum.com/.

[4] Q-Lambda, Getingevägen 46, 222 41 Lund, Sweden

LOW CURRENT MEASUREMENTS AT ALBA

Julio Lidon-Simon, David Fernandez-Carreiras, Jose-Vicente Gigante, Jerzy Jan Jamroz, Jorg Klora, Oscar Matilla, CELLS-ALBA Synchrotron, Cerdanyola del Vallès, Spain

Abstract

High accuracy low current readout is an extensively demanded technique in 3rd generation synchrotrons. Whether reading from scintillation excited large-area photodiodes for beam position measurement or out of gold meshes or isolated metallic coated surfaces in drain-current based intensity monitors, low current measurement devices comprise an ubiquitous need both for diagnostics and data acquisition in today's photon labs. In order to tackle the problem of measuring from various sources of different nature and magnitude synchronously, while remaining flexible at the same time, ALBA has started a project to develop a 4 independent channel electrometer. It is based on transimpedance amplifiers and integrates high resolution ADC converters and an Ethernet communication port. Each channel has independently configurable range, offset and low pass filter cut-off frequency settings and the main unit has external I/O to synchronize the data acquisition with the rest of the control system.

INTRODUCTION

With around 110 channels distributed in 7 beamlines, current measurement based diagnostics are the most common in Alba photon labs.

The most extended method for low current readout relays on range-configurable transimpedance amplifiers that convert the currents generated in the different sources into voltages. Those voltages are later read out either by ADCs or converted to pulses by Voltage to Frequency Converters (VTFs) and counted in counter cards. Lately, also some solutions based on charge integrators are appearing in the market, but transimpedance amplifiers are still the most common battle horse when it comes to low current measurement.

The dynamic range of the aforementioned currents spans from 1mA to few pA with bandwidths below a few kHz.

Apart from the characteristics of the current signal, it's important to mention that a usual problem coming up while measuring such low currents is the appearance of noise coupled, most often, from the power network, either through the cables or any other element of the setup. While noise coupling reduction techniques like proper shielding are the most effective solution, preventing the noise appearance, one desirable feature of the subsystem in charge of the measurement, especially for slow varying signals, is the possibility of selecting low band pass filters that attenuate out-of-band noises.

The beamline control system reads these currents with acquisition times typically in the range of 1ms to 1s and quite often the acquisition window must be precisely synchronized with other elements in the lab environment.

The different diagnostic equipment come with channels arranged in different configurations, from typical single channel intensity monitors to dual or quad-channel in 1D or 2D X-ray beam position monitors.

Beyond pure current measurements, beamline scientists often face the need of automate the compensation of low-frequency drifts due to thermal load changes or mechanical instability in optical components as mirrors or monochromators. To do that, an output signal to steer piezoelectric actuators or stepper motors can be generated after processing the signal acquired from photodiodes or drained out of isolated metallic surfaces exposed to the X-rays.

In order to provide us with a general platform to confront all those applications having as common link the measurement of low currents the Electronics section in Alba Computing Division launched the AlbaEm project.

The main target of this project is to produce a modular, transimpedance amplifier based, 4 independent channel electrometer with Ethernet communication with the possibility of accepting I/O expansion boards.

In the next paragraphs we detail the characteristics of the designed solution and the current status of the project.

SYSTEM ARCHITECTURE

The system architecture was chosen to maximize modularity to simplify at the same time future upgrades of the different subsystems, troubleshooting and maintenance.

The platform can be divided in the following main parts: Microprocessor board, current amplifiers (up to 4 units), main, I/O board and front panel board (see Fig. 1).

Current Amplifiers

With a first typical bipolar transimpedance amplifier layout stage and a second voltage gain stage, the amplifier provides gains from 1E+4 to 1 E+10 V/A for ±10V output in 8 full ranges from 1mA to 100pA. In the first stage, specific circuitry provides input offset voltage adjustment down to 1μV.

This first part feeds a second order low pass filter with 3dB cut-off frequency configurable to 100Hz, 10Hz and 1Hz. The filter can also be bypassed.

Finally, an output buffer stage allows polarity inversion and feeds both the upstream internal ADCs and a rear panel connector.

The analog ground is isolated from the equipment chassis to prevent ground loops and all the configuration is SPI based and stored in an onboard FLASH memory.

Microprocessor Board

The Electronics section in Alba Computing division has made extensive use of Rabbit microprocessor MiniCores

[1] in the past. These modules consist of small form factor PCBs with an embedded Rabbit microprocessor, flash and RAM memories and ethernet PHY. The microprocessor has a CISC arquitecture and supports different peripherals.

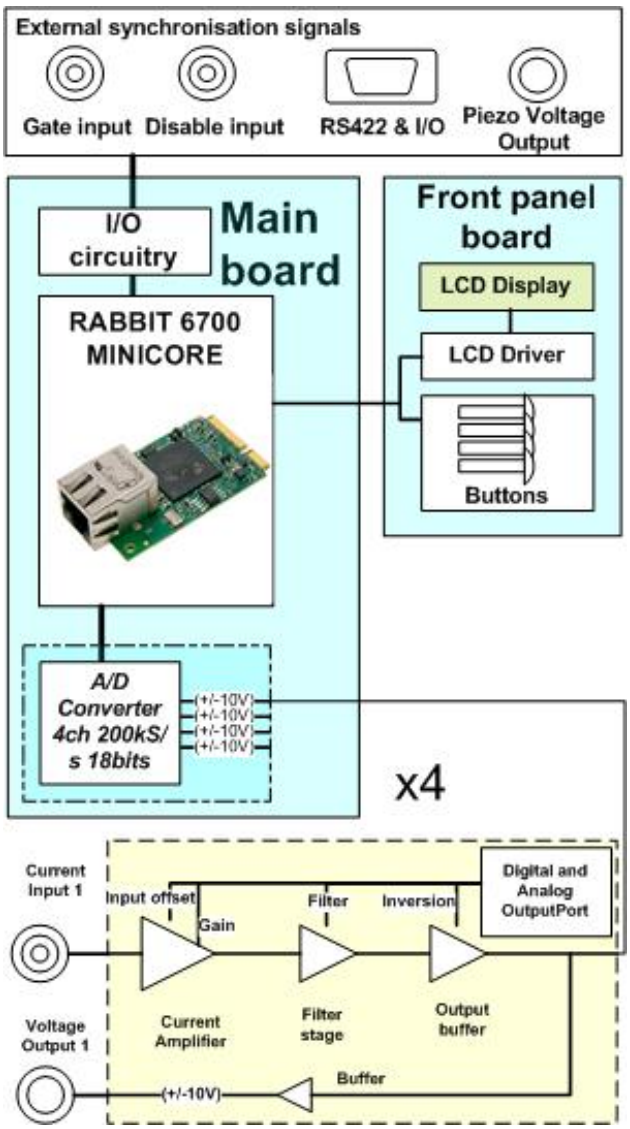


Figure 1: General system architecture.

The accumulated know-how and all the libraries developed for previous projects made the choice of one of these minicores the best option to minimize development time.

In the chosen MiniCore a 200MHz microprocessor is the central module. It has 1MB of RAM that can be used for data storage purposes. The microprocessor cares about the control and configuration of the rest of the subsystems either through SPI or its 8-bit external parallel bus. Communication is achieved via the onboard 10/100Mb Ethernet port.

Main Board

The main board hosts the rest of the boards and the ADCs and all the necessary gluing circuitry. Steered from the microprocessor board, a fast quad-channel 18bit bipolar ADC that can be run at sampling rates up to 200kHz carries out the data acquisition. With 1MOhm impedance inputs, an onchip 2nd order antialiasing filter and a configurable sinc digital output filter that helps to improve SNR and reduce data throughput unnecessary for low bandwidth, it fits perfectly the needs of an application like this.

I/O Circuitry

This subsystem contains the circuitry to adapt digital signal levels between the microprocessor and the exterior both for the synchronisation input signals and the output signals generated in applications where regulation of external actuators from current input signals is necessary.

Front Panel

Despite the electrometer is meant to be fully controlled remotely, a basic user interface will be available at the front panel so that it can be used as bench-top unit if necessary. The panel has an attached board carrying the minimum electronics to steer an LCD display and interface buttons to the microprocessor.

PROJECT STATUS

The first amplifier prototype was designed in November 2010 and first tested in January 2011. Thanks to the good results of the first amplifier prototype, the facility gave thumbs up to the project one month later. It was decided that it would be produced a number of electrometers for the beamline commissioning in October this year. Due to the strong schedule constraints, it was decided to mount in those electrometers a microprocessor and a 12bit ADC already used in another project. This board would be replaced later by the final one in the elements were the extra resolution would be required.

In the next paragraphs are described the performance of the designed amplifier and the acquisition board used for this first version of the device (see Fig. 2).

Figure 2: Low resolution electrometer rear panel.

Amplifier

The amplifier design has successfully reached the final stage. With up to 5 decades of noise free dynamic range, 3.2kHz bandwidth (1.5kHz for the last three ranges), filters providing extra out-of-band noise rejection, input offset voltage reduction down to 1uV (optimum for reading pA currents from large PIN diodes without saturating the amplifier output) and low drifts, it provides a very good foundation to add all the planned electrometer functionality (see Table 1).

In Table 1 are displayed the noises measured at the output of the amplifier and the correspondent SNR for each range and three different filter configurations. They were measured at constant input current of 10% of each range and a 20bit ADC. The output RMS voltage noise measured has been converted to current units to simplify the table interpretation.

Control and Signal Acquisition Boards

The work on the first prototype of the fast ADC main board is ongoing. In parallel, and based on a development for a previous inhouse project, the microprocessor and a 12bit ADC have been used to produce a lower resolution electrometer. This version, has been designed keeping it compatible with an upgrade to the future high-end ADC. That upgrade will just need the replacement of the main board with the one containing the new ADC, keeping chassis, amplifiers, microprocessor and firmware (with the evident exception of exchanging the low level library accessing the ADC by the new one).

Table 1: RMS Output Noise in Current Units and SNR for the Different Range and Filter Configurations

	No filter		10Hz		1Hz	
Range	**RMS**	**SNR(dB)**	**RMS**	**SNR(dB)**	**RMS**	**SNR(dB)**
1mA	7nA	103	7nA	103	7nA	103
100µA	700pA	103	700pA	103	700pA	103
10µA	300pA	90	70pA	103	70pA	103
1µA	50pA	86	7pA	103	7pA	103
100nA	4pA	88	800fA	102	700fA	103
10nA	700fA	83	150fA	96	80fA	102
1nA	500fA	66	100fA	80	20fA	94
100pA	500fA	46	100fA	60	20fA	74

Performance

This model has a limited sampling rate of 1kHz and a maximum bandwidth of 500Hz (due to the presence of the antialiasing filter). While the 12bits resolution of this electrometer version's ADC and its internal noise are an evident performance penalty, its use as diagnostics according to commissioning plan will be most often for acquisitions in the 0.1s range where averaging improves the total system noise.

The total ADC subsystem noise was found to be 7mVrms (for a ±10V span) that averaged by 10 or 100 samples (acquisition times of 10ms or 0.1s for the fixed 1kHz sample rate) leaves good 4 decades of noise-free current measurement in each range (provided that the ADC is the noise limiting element for that range).

Functionality

In order to ease the integration of the electrometer in the beamline control system, the following features have been implemented:

- Digital filter averaging: A size configurable moving average filter can be enabled to low-pass filter the digitized currents.
- Buffered measurements: The electrometer saves in its memory a sequence of a software predefined number of acquisitions. This acquisition "points" are available to the beamline control system as soon as they are acquired. It is not necessary to wait till sequence acquisition end to get the values.
- Software triggered acquisitions: The electrometer averages current readouts at the maximum sampling rate during a preconfigured time window after a command arrival.
- Hardware triggered acquisitions: A TTL trigger signal accessible from the rear panel starts an acquisition for a preconfigured time window
- Hardware gated acquisitions: The TTL signal defines the beginning and the end of the acquisition time window.
- Events: A list of clients can be provided by software. When the electrometer finishes an acquisition it sends to those clients the acquired values via UDP. This feature reduces the need of polling and the amount of network traffic.
- Remote firmware update. New features can be added remotely via TFTP. This reduces the maintenance and firmware update times and simplifies the addition of new features.

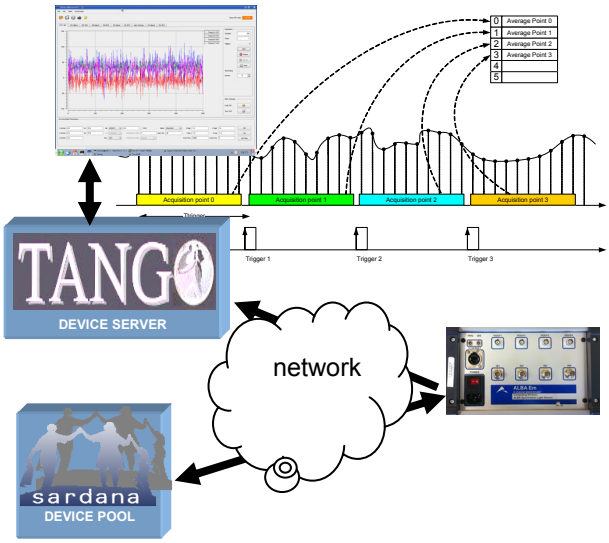


Figure 3: Integration in the beamline control system.

Integration in the Beamline Control System

The electrometer has been successfully integrated in SARDANA[2], the TANGO[3] based control system for instrumentation and data acquisition used at ALBA beamlines. In the current architecture, a python low level library to access the electrometer via UDP and a Sardana Pool Controller permit synchronization of measurements with other data acquisition in the beamline, while a TANGO Device Server acts as interface for GUIs (see Fig. 3).

SCHEDULE

The first amplifier prototype was designed in November 2010 and first tested in January 2011. In Fabruary of this year the project was finally approved and the first units with the 12 bit ADC were assembled and tested by May 2011. 25 electrometers were produced during summer for the beamline commissioning in October this year. During all this time the firmware has been steadily improved. The first fast ADC prototype is expected for February 2012. The implementation of functionality for optical equipment position regulation will come later.

CONCLUSIONS

AlbaEm is a living project. With very good results in the analog part of the design and an already mature functionality, it can fullfill the scientist requirements for the October 2011 commissioning at the Alba beamlines.

The amplifier is considered to be the final one for general purposes, though other versions could be built in case more specific applications arise.

An acquisition system based in a 12bit ADC used in a previous inhouse project has been designed, implemented and tested. The units are prepared to be updated with just one part replacement wherever extra resolution is necessary. A fast 18bit ADC is expected to be ready the first quarter next year.

In less than one year, the team has been able to put together 25 units of a robust system that fulfills the basic needs for current based diagnostics. Currently, these 25 units are already installed in the 7 beamlines and ready for the first photon beams.

Moreover the high potential of the new updates should convert AlbaEm in a platform that will allow us to tackle all the future needs of the scientists at Alba beamlines.

ACKNOWLEDGEMENTS

We want to thank P. Fajardo, H. González and R. Hino from the Detector and Electronics Group at the European Synchrotron Radiation Facility for sharing with us their knowledge on current measurements and transimpedance amplifiers. Without their help this project wouldn't have come to results so fast.

We also want to profit to these lines to thank A. Milán and G. Cuní in the Controls Section of ALBA Computing Division for their contribution in the integration of the electrometer in the contol system.

And last but not least, we want to thank to A. Camps and the ALBA Electronic Section technicians S. Astorga, X. Fariña, J. Pagés, A. Ruz and B. Saló that worked hard this year in order to meet all the deadlines.

REFERENCES

[1] Rabbit minicores, from DIGI International http://www.digi.com.

[2] Sardana, the Tango based Instrumentation and Data Acquisition Control System, http://computing.cells.es/services/collaborations/sardana.

[3] Tango. The object oriented distributed control system, http://www.tango-controls.org.

THE TIMBEL SYNCHRONIZATION BOARD FOR TIME RESOLVED EXPERIMENTS AT SYNCHROTRON SOLEIL

J-P. Ricaud*, P. Betinelli-Deck, J. Bisou, X. Elattaoui, C. Laulhé, P. Monteiro, L. S. Nadolski, G. Renaud, S. Ravy, M. Silly, F. Sirotti, Synchrotron SOLEIL#, Saint Aubin, France

Abstract

Time resolved experiments are one of the major services that synchrotrons can provide to scientists. The short, high frequency and regular flashes of synchrotron light are a fantastic tool to study the evolution of phenomena over time. To carry out time resolved experiments, beamlines need to synchronize their devices with these flashes of light with a jitter shorter than the pulse duration. For that purpose, Synchrotron SOLEIL has developed the TimBeL board fully interfaced to TANGO framework. This paper presents the main features required by time resolved experiments and how we achieved our goals with the TimBeL board.

INTRODUCTION

In synchrotron facilities, bunches of electrons turn inside a storage ring. Each time a bunch passes in front of a beamline, the beamline receives a flash of intense light. These light pulses are used by scientists to study phenomena evolving over time.

The short duration of the flashes allows the experiment scientist to freeze the current state of the sample. By analogy with photography, when the exposure time or the flash duration is too long, a moving subject will appear fuzzy on the picture (Figure 1). If the exposure time or the flash time is reduced, the subject appears clearly.

Figure 1: Fuzzy picture due to long exposure.

Thanks to the circular storage ring of a synchrotron facility, the light flashes occur continuously and at the revolution frequency. Using this stroboscopic effect, it is possible to study the evolution of a sample over a long period of time.

At Synchrotron SOLEIL, three main bunch filling patterns are dedicated to temporal studies: single bunch, eight bunch and hybrid modes (Figure 2). The associated flash frequencies are 847kHz for the single bunch and the hybrid modes, and 6.77MHz for the eight bunch mode. The flash duration is between 20ps to 40ps RMS. A low alpha mode to reduce this time down to 4ps will be proposed to users by the end of the year.

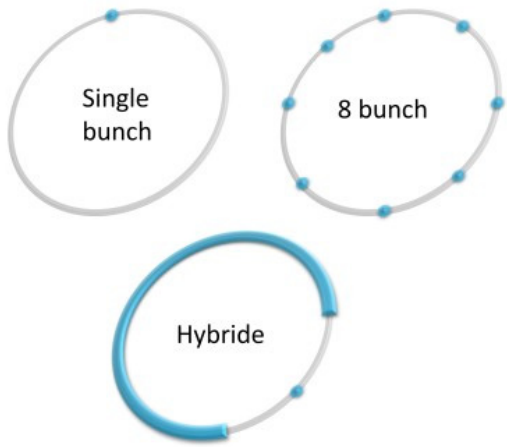


Figure 2: Hybrid, single and 8 bunch filling mode.

REQUIREMENTS

Synchrotron light flashes are not sufficient for scientists to perform time resolved experiments. They also need to synchronize their devices with bunches of electrons flying inside the storage ring, and delays must be added to compensate for cable and device time offsets (Figure 3).

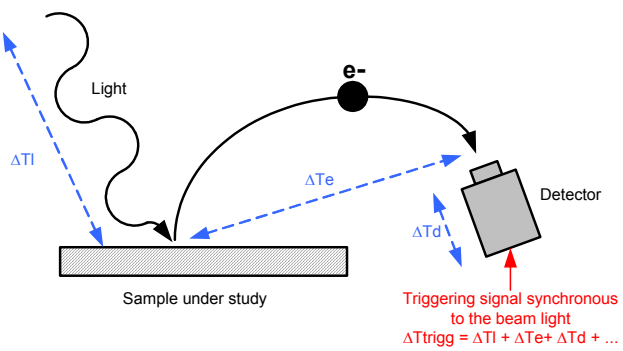


Figure 3: Delay offsets.

Because there are no dedicated pickup electrodes at Synchrotron SOLEIL, the synchronization triggers are generated from the radio frequency clock (352.196MHz). To guarantee synchronization with each bunch of electrons, the jitter of the system must be less than the bunch length (20ps RMS for the normal operating mode).

* jean-paul.ricaud@synchrotron-soleil.fr
# http://www.synchrotron-soleil.fr

As a first step a survey was performed to determine the needs of ten beamlines. A requirement matrix was made based on these inputs. It led us to design a system with: two independent outputs. Each output should provide a frequency between 88MHz and 0.76Hz. Each output should be delayed by step of #10ps between 0 and 1.44µs. This is enough to cover the full revolution period of the storage ring (1.18µs). The high / low level ratio of the pulse should be adjustable by the user. Clock outputs should be automatically synchronized to the storage ring clock. This allows us to always have the same time offset with respect to the electron bunch each time the machine is filled with a temporal bunch filling pattern. All these features must be user configurable through the control system.

To fulfill the above requirements, we have developed the TimBeL (TIMing BEamLines) board.

THE TIMBEL BOARD

The TimBeL system is a compact PCI board. It is made of a mother with one daughter board (Figure 4). All functions are performed inside a FPGA (Field Programmable Gate Array) implemented on the mother board. A PLX Technology [1] chip is used to communicate with the compact PCI crate. To enable experiments to remain always synchronous with the same bunch of electrons, the storage ring clock (CLK_SR) and the radio frequency clock (CLK_RF) are provided by the machine to beamlines. These clocks are used inside the FPGA as main clocks for state machines.

Because the jitter is too large on the FPGA outputs, a daughter board with a jitter cleaner has been added to the system. This board also provides delay lines for compensating time offsets by 10ps steps.

This architecture is versatile. Both the mother board and the daughter board may be upgraded independently to improve performance or to meet new requirements. Considerable work has been done on the PCB (Printed Circuit Board) to ensure good signal quality and integrity on both boards.

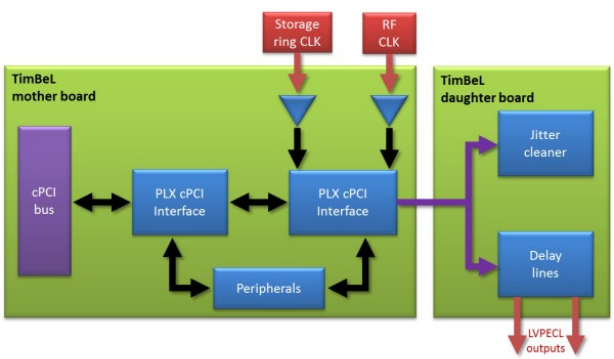


Figure 4: TimBeL architecture.

The mother board may also be used with other types of daughter boards for applications other than beamline synchronization.

A Tango device is used to configure the board outputs frequencies and time delays. This device accesses the FPGA's registers through a low level driver developed at Synchrotron SOLEIL (Figure 5).

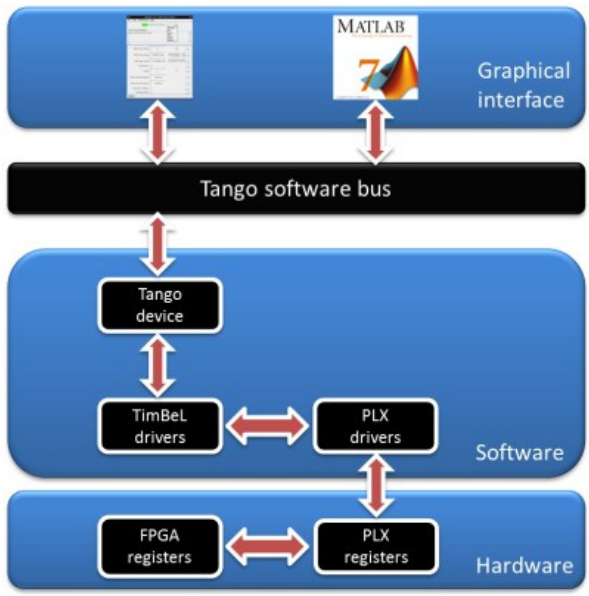


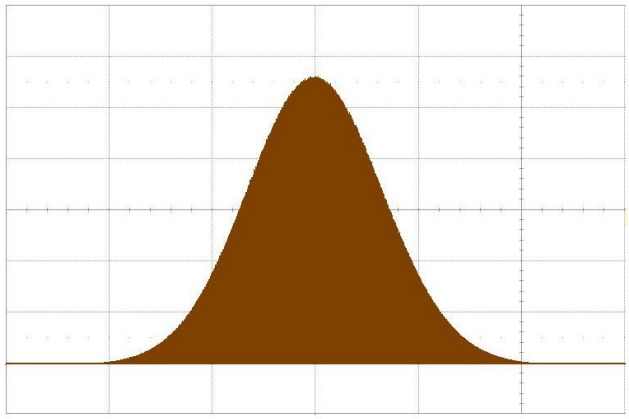
Figure 5: Software architecture.

Mechanical parts of the system have also been designed by Synchrotron SOLEIL. Manufacturing of the boards is however entrusted to an external subcontractor.

JITTER PERFORMANCE

The major concern with the performance of the board was jitter. To remain synchronous to the electrons bunches, the jitter must be lesser than 20ps RMS and we have totally achieved this goal.

The differential LVPECL outputs have a jitter of 3.3ps RMS @ 6.77MHz which is the 8 bunch frequency (Figure 6). In single ended mode, a LVPECL output has a jitter of 3.8ps RMS @ 6.77MHz. The source jitter (R&S signal generator [2]) is about 7ps RMS. All measurements have been made with a LECROY SDA 820Zi-A oscilloscope [3].



Jitter Kit	Period@level	Freq@level	Cycle to Cycle
mean	147.64257 ns		-2 fs
sdev	3.31 ps		5.73 ps
pkpk	29 ps		50 ps
max dev+	14 ps		23 ps
max dev-	-15 ps		-27 ps
max dev	15 ps		27 ps
num	101.493e+3		67.644e+3
status	✔		✔

Figure 6: TimBeL jitter.

Although it was not designed for this purpose, an Input / Output port on the board is also frequently used by beamlines as it directly provides a single ended LVTTL clock. On this I/O, the jitter is about 24ps RMS, measured with a TEKTRONIX TDS6640B oscilloscope [4].

SOME RESULTS

Each year, a few weeks are dedicated to time resolved experiments. In 2011, Synchrotron SOLEIL has provided its users with 2 single bunch and 2 eight bunch runs. Already 4 weeks of hybrid operation have been allocated and for the commissioning of the pump-probe lasers on TEMPO and CRISTAL beamlines this mode will be delivered until the end of the year.

Since 2009, a total of 6 beamlines have been using the TimBeL board either with their own devices or with the devices of their users. To date, all users have been satisfied by this board and its functionalities. No major complaints or problems have been reported.

Among other experiments, during last month, CRISTAL beamline has used this board to study photoinduced spin transition in molecular crystal [5]. The TimBeL was used to trigger a laser, a shutter, and an XPAD3.2 [6] synchronously with electron bunches inside the storage ring.

At the TEMPO beamline pump probe experiments were performed to directly observe the effect of Oersted fields in nanostripes during spin injection [7]. The TimBeL board is currently used for time resolved photoelectron spectroscopy experiments [8] and will be an essential element for planned experiments which will couple fs-laser pulses with the soft X-rays coming from the beamline.

The TimBeL board was also used successfully to test the pseudo single bunch mode under study at Synchrotron SOLEIL [9]. In this special hybrid mode, an isolated bunch of electrons is placed on a different closed orbit with respect to other bunches, using a fast pulsed corrector, active only for the isolated bunch. A TimBeL board was used on the machine to trigger the fast kicker magnet at a frequency of 1kHz. Another board was used on the TEMPO beamline to trigger the Scienta SES 2002 synchronously to this bunch. The first results obtained at the end of 2010 were very promising. This is an innovative and on-going in-house development.

CONCLUSION

A few years after the commissioning of the machine synchronization system [10], Synchrotron SOLEIL provided to beamlines the TimBeL board for time resolved experiments.

Unlike the machine timing system, the TimBeL board has been entirely designed by Synchrotron SOLEIL. This has helped us to modify and adapt it to experiments requests several times. Five firmware upgrades have been made since the board was installed on beamlines. Because we are perfectly familiar with this board, it was possible to adapt it very quickly, in just a few days, to new experimental needs and new data acquisition procedures.

The TimBeL board is a low cost and versatile system that has given satisfaction to beamlines for time resolved experiments.

REFERENCES

[1] PLX Technology, http://www.plxtech.com.
[2] Rohde&Schwarz, SML01, 9kHz ... 1.3GHz.
[3] LeCroy, WaveMaster 8 Zi-A oscilloscope, 20GHz, 4x40GS/s.
[4] Tektronix, TDS6640B oscilloscope, 6GHz, 20GS/s.
[5] C. Laulhé et al., "Time-resolved pump-probe diffraction experiment at SOLEIL synchrotron: photoinduced spin transition in the molecular crystal [TPA Fe(III) TCC] PF6", PIPT4 2011.
[6] K. Medjoubi et al., "Performance and application of the CdTe- and Si-XPAD3 photon counting 2D detector", Journal of Instrumentation 2011, 6(1):C01080.
[7] V. Uhlir et al., Physical Review B 2011, 83(2): art. n 020406.
[8] N. Bergeard et al., "Time-resolved photoelectron spectroscopy using synchrotron radiation time structure", Journal of Synchrotron Radiation 2011, 18(2):245-250.
[9] L.S. Nadolski et al., "First measurements with a kicked off axis bunch for pseudo single bunch mode studies at SOLEIL", Proceeding of IPAC, San Sebastian (2011).
[10] J-P. Ricaud et al., "Synchronization System of Synchrotron SOLEIL", ICALEPCS, 2007.

THE RF CONTROL SYSTEM OF THE SSRF 150 MEV LINAC

S.M. Hu, J.G. Ding, G..Y. Jiang, L.R. Shen, M.H. Zhao, S.P. Zhong
SSRF, SINAP, Shanghai 800-204, P. R.China

Abstract

Shanghai Synchrotron Radiation Facility (SSRF) use a 150 MeV linear electron accelerator as injector, its RF system consists of many discrete devices. The control system is mainly composed of a VME controller and a device interface. The device interface is a home-made signal conditioner with DC power supplies. The DC power supplies are used for driving the mechanical phase shifters. This uniform signal conditioner serves as a hardware interface between the controller and the RF components. The control software is based on EPICS toolkit. Device drivers and related runtime database for the VME modules were developed. The operator interface was implemented by EDM.

INTRODUCTION

The injector of the Shanghai Synchrotron Radiation Facility (SSRF) is a 150 MeV linear electron accelerator [1], and it is shown in Figure 1. The RF system of the 150MeV Linac is composed of many individual components with different interfaces. The layout of RF system is shown in Fig. 2. A universal RF device interface was designed and manufactured to accomplish the RF system control mission. Connections between the local controller and RF devices are fully isolated by either the controller interface or the devices itself.

The RF controller consists of a VME system and a device interface. It controls many of the discrete RF components directly, that include 3 solid RF amplifiers, 12 klystron focus coil Power supplies, 3 phase shifters, and it also monitors vacuum of the klystrons and operates the reverse power protect resetting keys. Besides, there are several sub control systems:

- A sub-harmonic cavity is tuned by a stepping motor, which is controlled through a serial communication server.
- A Phase control sub-system [2] consist of a compact PCI cabinet and several modules.
- The controls for the 2 sets modulators [3] of RF power system by PLCs.

The Linac RF control system has been put into operation since July of 2007 and worked stably by now.

Figure 1: 150MeV Linac of SSRF.

SYSTEM CONFIGURATION

The Linac RF control system employs Ethernet-based distributed architecture. In the RF control system, we adopt the Experimental Physics and Industrial Control System (EPICS). The RF control system consists of

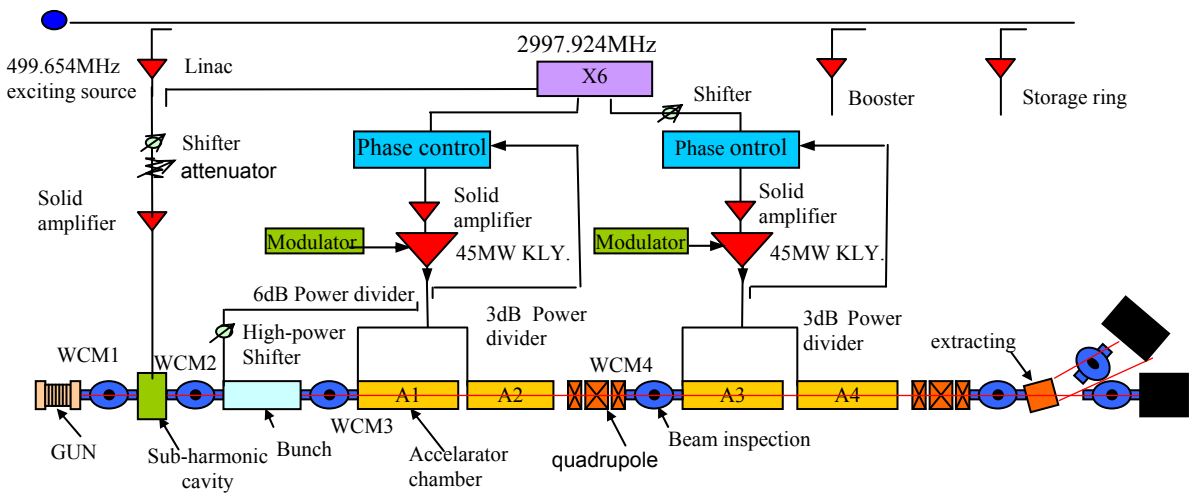


Figure 2: 150MeV Linac RF system layout.

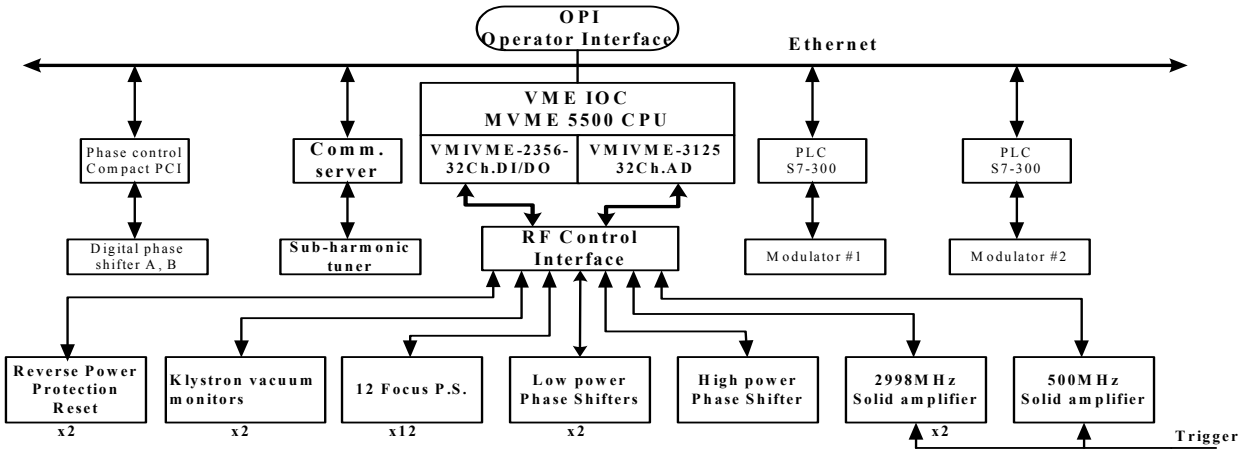


Figure 3: RF control system structure.

Operating Interfaces (OPI), Input/Output Controller (IOC). The Gigabit Ethernet switches link the OPI and IOCs to the SSRF control net. The SSRF timing system provides triggers to the amplifiers and modulators. The structure of RF control system is shown in Figure 3

Controller and Interface

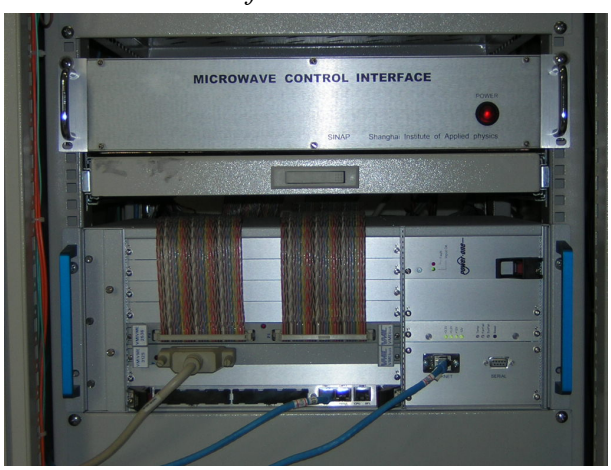


Figure 4: IOC controller and Interface.

The RF Controller and interface are shown in Figure 4. It consist of a 9-slot VME crate, a MVME 5500 CPU board, a VMIVME 2536 digital I/O isolated interface board and a VMIVME3125 A/D isolated interface board.

The dedicated interface of RF devices is designed and manufactured by SSRF for the linac RF control system. It is used not only for interface conditioning, but for power supply as well. The interface is designed for universal purpose, its most configurations are jumper selectable. The input/output interface between the controller and devices are optoelectronic isolated.

Software

EPICS toolkit was used for development of the RF control system. For different equipment interface and communication protocol, specific device support and driver is necessary to make the device to be controlled by EPICS [4]. The VME device support and drivers adopted in the RF control system were developed. The epics driver for VMIVME 2536 was written in 2002, for a 100MeV Linac control system; and epics driver for VMIVME 3125 was written in 2006. There are some drivers provided by the EPICS community, for example, S7-PLC [5] and NetDev [6] are used for Ethernet device control.

The operator interfaces of the RF control system were implemented by EDM.

CONCLUSION

The RF control system of SSRF 150MeV Linac has been operating for a period of more than 4 years, it has been testified that this control system is a low cost and more cost-effective solution to SSRF linac RF system. The control interface is a universal and convenient adapter to the controller and RF devices, and it works stably.

REFERENCES

[1] Zhao Minghua *et al*. "Commissioning of the 150 MeV SSRF Linac", Proceedings of EPAC08, Genoa, Italy.

[2] Yin Chongxian *et al*. "Phase control system for SSRF linac", Nuclear Science and Techniques 19 (2008) 129–133.

[3] Yuan Qibing *et al*. "Design of 110MW PLC-based pulse modulator controller", SINAP, Nuclear Techniques Vol.30, No.10 October 2007.

[4] Kraimer M R, Anderson J, Johnson A, *et al*. EPICS: Input/Output controller application developer's guide Release 3.14.8, 2005.

[5] Dirk Zimoch, "S7plc EPICS driver documentation." March 2005.

[6] Jun-ichi Odagare, NetDev, SSRF modified Version.

ONLINE EVALUATION OF NEW DBPM PROCESSORS AT SINAP *

Y. B. Leng #, X. Yi, L. W. Lai, G. Q. Huang, Y. B. Yan, SINAP, Shanghai, China.

Abstract

In this paper, we report our online evaluation results for new digital BPM signal processors, which are developed for the SSRF and the new Shanghai SXFEL facility. Two major prototypes have been evaluated. The first algorithm evaluation prototype is built using commercial development toolkits modules in order to test various digital processing blocks. The second prototype is designed and fabricated from chips level in order to evaluate the hardware performances of different functional modules and assembled processor.

INTRODUCTION

Digital beam position monitor (DBPM) processor is world wide used in light sources and FEL accelerators [1-3]. Few microns resolution at turn-by-turn (TBT) rate and sub-micron resolution at slow application (SA) rate are achieved for circular accelerators with commercial product [4].

In order to accumulate technologies and enhance capability of instrument development two prototypes of DBPM processors have been initiated since 2008 at SINAP. One prototype is based on commercial ADC evaluation board (ICS1554) to test various signal processing blocks and EPICS IOC interfaces [5]. The other prototype is custom designed and built from chips level to be a hardware frame of processor [6]. The primary target of this processor is SSRF storage ring. But equipped with proper RF front-end this processor could be used for the new Shanghai SXFEL facility. Both prototypes have been completed and passed lab test recently. After that these two prototypes merged together to form a final version ready for online evaluation.

The practical beam signal is totally different with sine wave produced by RF signal generator in lab. Online evaluation has to be performed to verify the functionality and performance of the new DBPM processors. A spare button-type BPM sensor (C15BPM8) in the SSRF ring, which delivering the same signal as other BPMs, is used for this purpose. All function blocks including turn-by-turn, fast application (FA) and slow application are loaded into the new processor to acquire real beam data.

SPATIAL RESOLUTION EVALUATION

500k samples of TBT data, 500k samples of FA data and 3k samples of SA data are acquired to evaluate the spatial resolution of BPM processor. Figure 1 shows the histogram of these data in the vertical plane at 200 mA. Since the distribution of position readings is the combination of processor electronics noise and real beam orbit noise, the actual resolution should be better than σ. That means the TBT resolution of the new DBPM processor is better than 1.4 μm, FA resolution is better than 0.38 μm, and SA resolution is better than 0.21 μm.

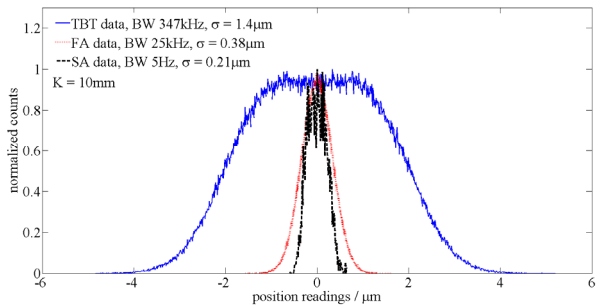


Figure 1: Spatial resolution evaluation online.

Non-Gaussian distribution of TBT data comes from a narrow band noise, which frequency is few tens kHz, produced in the DDC stage. The next optimization will focus on this.

Figure 2 shows the result of TBT resolution evaluation with different beam current. Resolution of 2 μm can be achieved when beam current is larger than 100 mA.

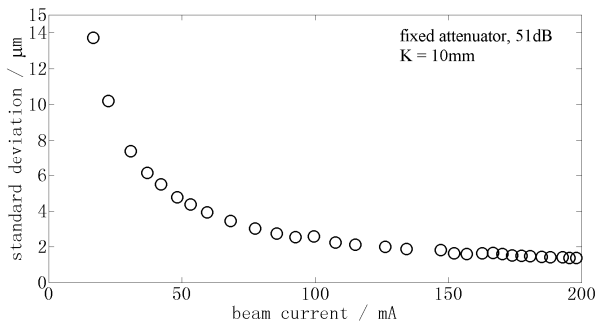


Figure 2: TBT resolution evaluation with different beam current.

TBT APPLICATION

The frequency domain analyze of TBT data is very powerful tools for machine study. All kinds of orbit noise which bandwidth lower than half of revolution frequency will show in the spectrum of TBT data and be identified easily. Combined with exciting kickers, fabric tune value, beta function and betatron oscillation phase advance can be derived from TBT data of multiple BPMs as well. Traditional DDC algorithm based TBT block has been implemented in the new BPM processor to demonstrate the above capabilities. Figure 3 shows the spectra of TBT data at vertical plane captured during injection.

*Work supported by 100 Talents Program of the Chinese Academy of Sciences and National Natural Science Fund (No. 11075198)
#leng@sinap.ac.cn

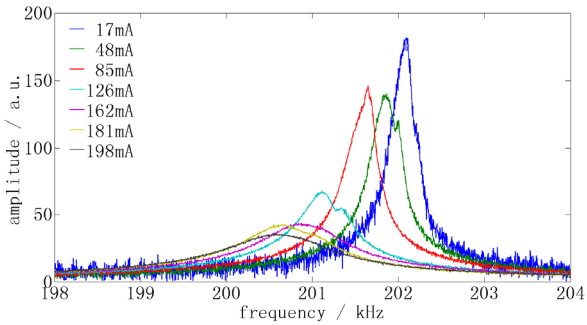


Figure 3: Beam spectra at different currents.

The buffer size as large as 500k samples provides Hz level frequency resolution. The tune shifts and betatron oscillation peak splitting during injection are easily observed. The information of beam impedance can be retrieved by fitting tune shifts with beam currents, shown in the Figure 4.

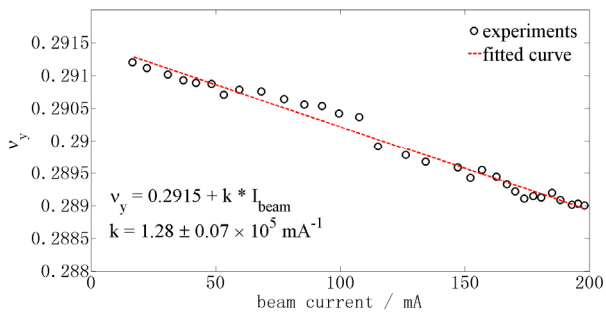


Figure 4: Betatron tune shifts during injection.

FA APPLICATION

Precise analyze of low frequency noise at TBT rate requires million samples of data. It will add too much CPU loading to processor to damage its reliability. Another choice is using fast application data (typical tens kHz) to do this job. In our case the data rate of FA block is configured to be 50 kHz and the buffer size is set to be 500k samples.

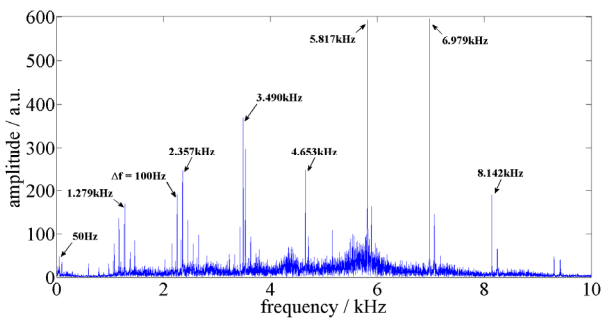


Figure 5: Orbit noise below 10 kHz.

Figure 5 shows the major horizontal orbit noise below 10 kHz. Low Q energy oscillation peak (5.8 kHz) and high Q electronics noise (1.279 kHz, 2.357 kHz, 3.490 kHz, and so on) coming from LLRF are easily observed. More optimization work of RF system can be done based on this analyze.

When we focus on the orbit noise below 100 Hz, the more details can be found shown in Figure 6. The central frequency of major part of vertical orbit noise is about 5 Hz. The horizontal orbit noise consists three parts: 1~2 Hz, 20 ~ 30 Hz and 40 ~ 60 Hz. The horizontal and vertical ratio of power line noise and its second harmonic indicate that the noise is probably coupled from magnet power supplies but not BPM processor. The more orbit optimization work can be done based on this analyze.

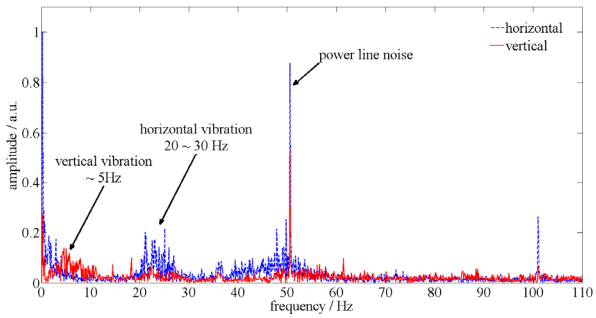


Figure 6: Orbit noise below 100 Hz.

SA APPLICATION

The basic functionality of SA block is providing close orbit information. With good calibration method and good linearity of ADC module the sum signal of four channels also can be used for beam current and lifetime measurement.

Figure 7 shows the orbit variation before, during and after injection captured by new DBPM processor. The horizontal orbit disturbances due to injection kicking during injection and the orbit changes due to close orbit correction after injection are clearly recorded.

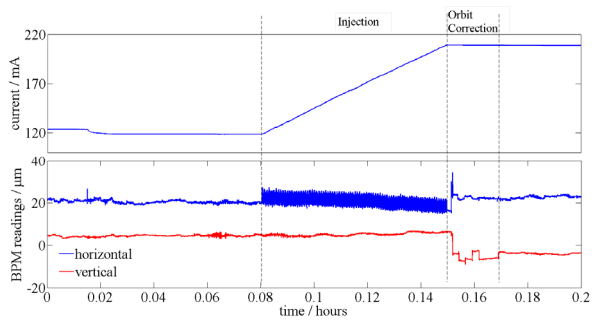


Figure 7: Measured orbit variation during injection stage.

The new processor shows very good linearity which makes measurement of beam current using sum signal possible.

Figure 8 shows the comparison of measured beam currents by DCCT and DBPM. The measurement error of DBPM due to nonlinearity is smaller than 1 mA in the range of 100 ~ 200 mA. This performance could be improved when high order calibration method applied.

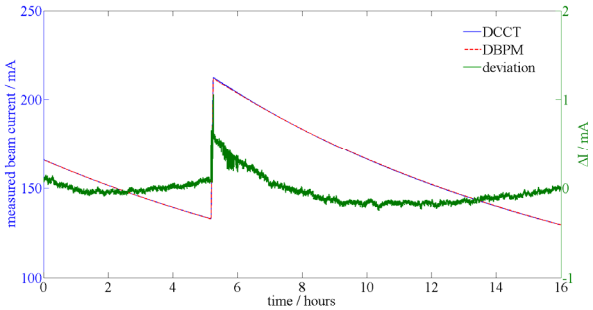


Figure 8: Comparison of measured beam currents by DCCT and DBPM.

Figure 9 shows the lifetime measurement result using DCCT and DBPM. It is obvious that the lifetime resolution of DCCT is much better than DBPM. But the traces of two measurements match together perfectly. The lifetime resolution of DBPM can be improved by involving more BPMs.

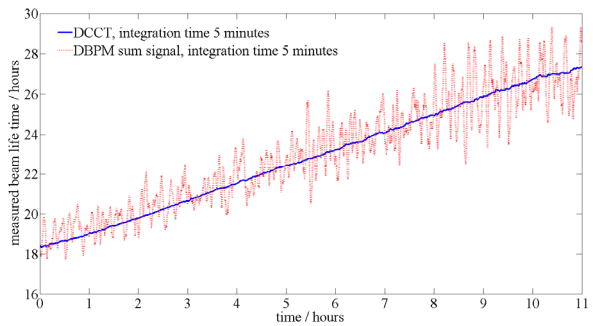
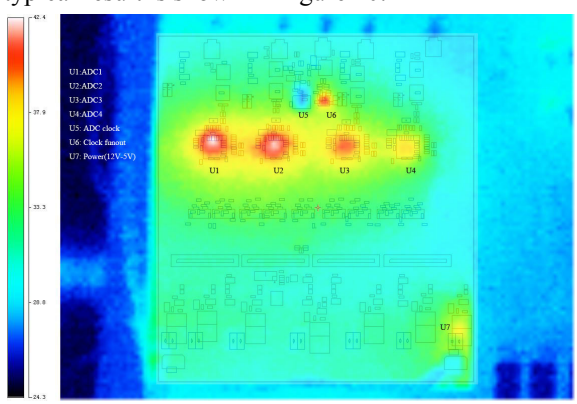


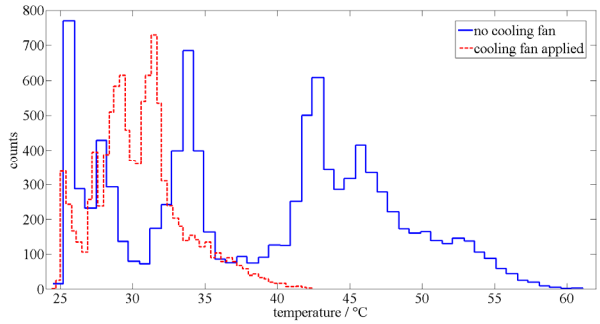
Figure 9: Beam lifetime measurement using DBPM sum signal.

THERMAL LOADING INVESTIGATION

Thermal loading is a critical issue for high performance instruments. High thermal loading, large thermal gradient or large thermal drift of circuit could crash the precision, reliability and long-term stability of DBPM processor. In order to collect enough information for the next technical prototype a thermal imager is used to investigate the thermal distribution of RF and ADC modules in the field. A typical result is shown in Figure 10.



(a) Spatial distribution of thermal loading.



(b) Histogram of thermal loading.

Figure 10: Thermal loading investigation of ADC module.

It is obvious that ADC chips are major thermal producers and cooling fan is definitely needed. After cooling fan applied the temperature of ADC chips decrease from 60 degree down to 42 degree and the average temperature of PCB board decreases from 43 degree down to 32 degree. More precise cooling design will be made and the new lower power ADC chips will be used in the next technical prototype based on this investigation.

CONCLUSTION

A prototype of new digital BPM processor has been evaluated online at SSRF storage ring. Major signal processing blocks including TBT, FA and SA have been tested. Thermal loadings of RF and ADC module were investigated. Evaluation results show that the functionality and performance are comparable with commercial products and satisfied the requirements of operation and machine study. Evaluation results confirmed the current design. The next technical prototype will be built soon.

REFERENCES

[1] G. Rehm, M.G. Abbott, J. Rowland, etc., “Digital EBPMs at Diamond: Operational experience and Integration into a Fast Global Orbit Feedback”, DIPAC2007, Venice, Italy, 2007: 24-26.

[2] N. Hubert, L. Cassinari, J-C. Denard, etc., “The Soleil BPM and Orbit Feedback Systems”, DIPAC2007, Venice, Italy, 2007: 189-191.

[3] P. Leban, G. Decker, “Initial Tests of New Electron and Photon Beam Position Monitor Electronics at Advanced Photon Source”, DIPAC2011, Hamburg, Germany, 2011: 344-346.

[4] Y.B. Leng, K.R. Ye, W.M. Zhou, etc., “SSRF beam diagnostics system commissioning”, DIPAC2009, Basel, Switzerland, 2009: 24-26.

[5] L.W. Lai, Y.B. Leng, X. Yi, etc., “The Study and Implementation of Signal Processing Algorithm for Digital Beam Position Monitor”, PAC2011, New York, United States, 2011.

[6] X. Yi, Y.B. Leng, L.W. Lai, etc., “A Calibration Method for the RF Front-end Asymmetry of the DBPM Processor”, DIPAC2011, Hamburg, Germany, 2011: 56-58.

TEMPERATURE MEASUREMENT SYSTEM OF NOVOSIBIRSK FREE ELECTRON LASER

B.A.Gudkov, P.A.Selivanov, V.R.Kozak, E.A.Kuper, S.S.Serednyakov, S.V.Tararyshkin
BINP SB RAS, Novosibirsk, Russia

Abstract

This paper describes the temperature-monitoring system of Novosibirsk FEL. The main task of this system is to prevent the FEL from overheating and its individual components from damage. The system accumulates information from a large number of temperature sensors installed on different parts of the FEL facility, which allows measuring the temperature of the vacuum chamber, cooling water, and windings of the magnetic elements. Since the architecture of this system allows processing information not only from temperature sensors, it is also used to measure, for instance, vacuum parameters and some parameters of the cooling water. The software part of this system is integrated into the FEL control system, so measurements from all sensors are recorded to the database every 30 seconds.

INTRODUCTION

A high-power free electron laser (FEL), based on the microtron-recuperator[1], is under construction now at Budker Institute of Nuclear Physics. The first and second phases of the project were commissioned recently. The system for monitoring of the temperature of FEL components during its operation takes a very important place among other diagnostic systems of the FEL. The system consists of a large amount of temperature sensors installed on different parts of the FEL facility, including the vacuum chamber of the microtron-recuperator, the windings of the magnetic elements, and the elements of the water cooling system. The sensors are installed mainly at places with a hazard of overheating. They are also installed at places where temperature measured can give some useful information about the FEL operation, e.g. places of possible beam losses.

MEASUREMENT HARDWARE

The measuring part of this system consists of a few types of sensors (or some measuring devices) installed on different parts of the microtron-recuperator. Let us enumerate all these types and briefly describe the principles of their operation:

1. Temperature sensors. This is a small winding of copper wire enclosed in a metal case and toughly connected to the place the temperature of which is to be measured. The leads of this winding are connected to a special electric circuit. The output voltage of this circuit depends on the resistance of the winding. As the resistance of copper wire depends on its temperature and this temperature is actually equal to the temperature of the surface this sensor is connected to, the temperature could be calculated from the output voltage value. The calculation formula is as follows:

$$T = A * U + B \quad (1)$$

where A and B are constant coefficients and U is the output voltage.

2. Water sensors. These sensors are very similar to the temperature sensors, two large metal plates with a very small spacing used instead of a copper-wire winding . These sensors are installed on the floor of the accelerator hall. If water gets between these plates, the resistance of this circuit decreases noticeably, and the output voltage of the circuit also changes much. So, if the output voltage exceeds a certain value, one can suppose that there is water on the floor of the accelerator hall.

3. Vacuum “sensors”. These sensors are actually control devices for vacuum pumps. These devices measure the current of a vacuum pump; convert it to the output voltage, which could be measured by an ADC. These pumps are installed on different parts of the vacuum chamber of the microtron-recuperator, so collecting values of their currents could give information about the vacuum state of the accelerator chamber. The calculation formula for the pump current is as follows:

$$I = A * e^{B*U+C} \quad (2)$$

where A, B and C are constant coefficients, and U is the output voltage.

The output voltages from sensors of all the three types are measured by a few identical multi-channel ADCs connected to one CAN line and working in the same measurement mode. Thus, all the sensors of these three types are united into one subsystem of the common control system. The quantities of sensors for different stages of the FEL are shown in table 1.

Table 1: Quantity of Sensors for the 1st, 2nd and 3rd Stages of the FEL

Sensor type	FEL Stage		
	1st	2nd	3rd
Temperature	98	112	165
Water	3	3	3
Vacuum	14	19	25

The total scheme of this system is presented in Figure 1.

CONTROL HARDWARE

The choice of measurement devices is influenced by the following factors:

1. The accuracy of value to measure. Temperature sensors impose the highest requirements on the measurement accuracy. The measurement error of these sensors must not exceed 0.3 degree of Celsius. Since the coefficient *A* in formula 1 equals 300.0, the output voltage must be measured with an accuracy of 1.0 millivolt. So, the measuring device must have a less resolution.
2. The large quantity of identical sensors with readings to process. This factor makes one use devices with as many input channels as possible. It is also desirable that the device should support the multi-channel measurement regime. Application of devices with such characteristics would reduce the required number of devices and consequently the cost of the system.
3. The very low frequency of measurement. All above values – temperature, vacuum pump current, water presence – are parameters that change very slowly. For example, temperature changes noticeably only in tens of seconds. Therefore, a period for processing of readings of all sensors of five to ten seconds would be enough.

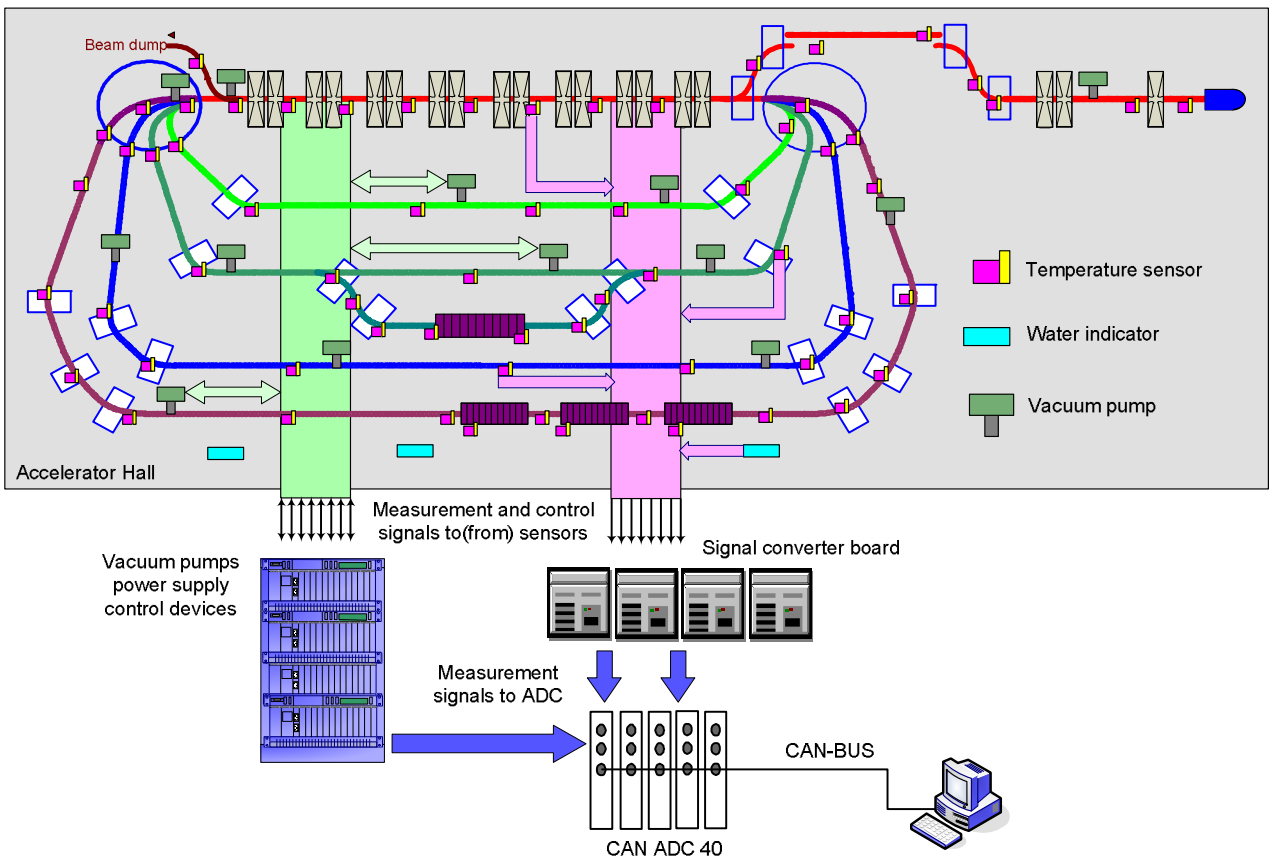


Figure 1: Main scheme of the system.

CANADC40 devices have been chosen for this system. These devices are 40-channel ADCs with the CAN-BUS[2] interface. They were designed and manufactured at BINP [3]. They meet all the above requirements.

Since different quantities of sensors are used for different FEL stages, the number of CANADC40 devices also varies (see Table 2).

Table 2: Quantity of CANADC40s and Channels Used for all FEL Stages

FEL Stage	CANADC40 Qty	Total channels	Used channels
1	4	160	115
2	8	320	144
3	8	320	193

CONTROL SOFTWARE

The control software for this system is a single application running on the IBM-PC computer. Communication with the CAN-BUS is realized with a CAN-to-Ethernet gateway developed at Budker INP. The main features of this application are as follows:

1. Representation of all sensors on a mnemonic scheme of the microtron-recuperator, which is displayed in the main window of the application. The position of the sensor in the scheme approximately corresponds to its position in the accelerator chamber. Every sensor is a small picture, which varies for different types of sensors. The color of the picture reflects the actual state of a sensor – *Normal, Not Connected, Warning,* and *Alarm*. The mnemonic scheme with sensors is shown in Figure 2.

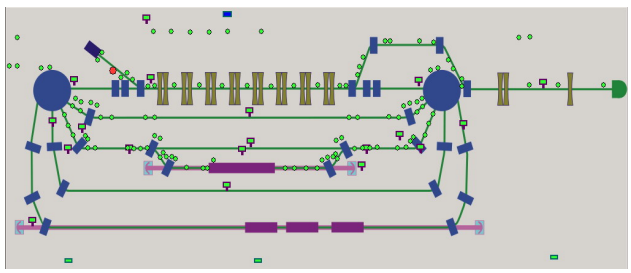


Figure 2: The mnemonic scheme of the microtron-recuperator.

2. Representation of values from all sensors as a row of vertical columns. In addition to the row of values, the second row contains the columns of time derivatives of the values of the sensors. The column height depends on the value of sensor or its derivative, correspondingly. Besides, the color of column reflects the state of sensor, as in the case of the sensor picture in the mnemonic scheme. A fragment of these rows is presented in Figure 3.

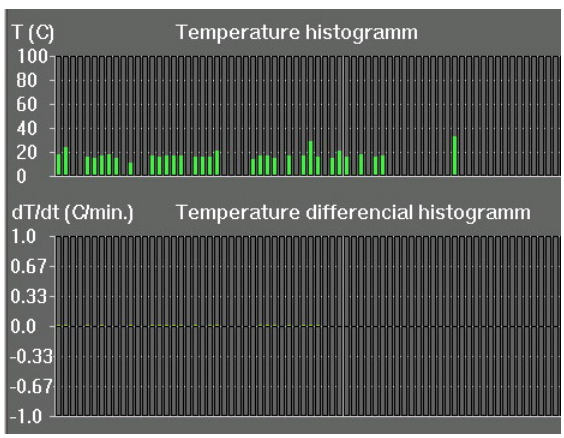


Figure 3: Temperature and derivative columns.

3. Flexible tuning of the configuration of the application. Using external configuration files, the user can specify a few different regimes of operation, which include different sets of sensors, different numbers of CANADC40 devices to be used, different pictures of mnemonic schemes and so on. This allows using one control application for all three stages of FEL operation. Depending on content of the configuration file and command-line parameter, the application operates in a corresponding FEL stage.

4. The Epics Channel Access Server. The server is built-in inside the application and runs in a separated thread. In the server, the Process Variables (PVs) represent the values and status of all sensors, processed in the actual operation mode. The PV name is connected with the name and type of a sensor in the application. The content of the Process Variables is used in a few high-level client applications. Besides, the Channel Archiver application writes down all values of Process Variables on the hard drive every 30 seconds. The stored history of the values of all sensors can be extracted for further analysis (see Figure 4).

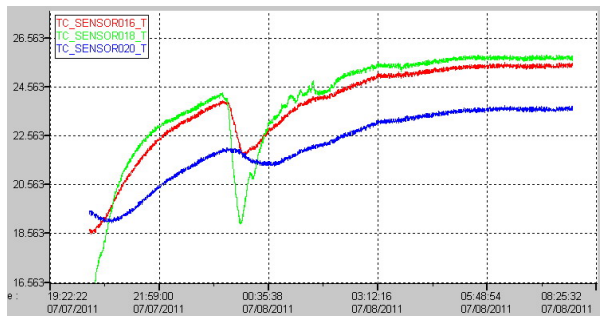


Figure 4: History of the values of a few temperature sensors.

CONCLUSION

The above-described system has been under operation for the first and second stages of the FEL for about 8 years. During the operation, the temperature measurement system demonstrated high reliability and ease of usage. The large quantity of sensors of different types and intelligent visualization of the collected information – values and their derivatives – give much information on the actual state of the FEL facility. The Epics Channel Access server in the program allows transferring data measured to any client application in the FEL control LAN.

REFERENCES

[1] N.A. Vinokurov et al. Novosibirsk free electron laser facility: Two-orbit ERL with two FELs. Proceedings of FEL2009, TUOD01.
[2] http://www.can-cia.org CAN In Automation, 2009.
[3] V.R.Kozak, E.A.Kuper. Microprocessor controllers for power sources. BINP preprint 2001-70, 2001.

TESTING DIGITAL ELECTRONIC PROTECTION SYSTEMS

S. Gabourin, A. Garcia Muñoz, CERN, Geneva, Switzerland

Abstract

This paper outlines the core concepts and realisation of the Safe Machine Parameters Controller (SMPC) test-bench, based on a VME crate and LabVIEW program. Its main goal is to ensure the correct function of the SMPC for the protection of the CERN accelerator complex. To achieve this, the tester has been built to replicate the machine environment and operation, in order to ensure that the chassis under test is completely exercised. The complexity of the task increases with the number of input combinations. This paper also outlines the benefits and weaknesses of developing a test suite independently of the hardware being tested, using the "V" approach.

INTRODUCTION

The SMPC ensures the correct configuration of the LHC machine protection system, and that safe injection conditions are maintained throughout the filling of the LHC machine. The SMPC receives information in real-time from measurement electronics installed throughout the LHC and SPS accelerators, determines the state of the machine, and informs the SPS and LHC machine protection systems of these conditions.

The SMPC is built to ensure reliability and availability of the transmitted information. The reception of the information and the generation of the state of the machine are done in complete redundant VME boards. An arbitration module then determines the correct information to send from the redundant information provided. This redundancy ensures high system dependability.

At the same time, redundancy makes calculations and the overall system more complex and subject to numerous errors or unexpected behaviours that would not be an issue for a non-redundant system. For this reason, a dedicated tester, the Safe Machine Parameter Tester (SMPT), has been developed to identify possible weaknesses of the system, and to validate the correct function of the SMPC.

SAFE MACHINE PARAMETERS

Components

The SMPC (see Figure 1) is composed of three types of electrical boards, Receivers (CISR), Generators (CISG) and an Arbiter (CISA) all based on the same basic PCB design, but having different functionality thanks to different VHDL code in FPGAs.

For the LHC, there are two redundant CISR (CISRA and CISRB), each one decoding a pair of energies and a pair of intensities. These CISR pass information to 2 redundant CISG (CISGLA and CISGLB).

The CISG use energies and intensities from the CISR, in conjunction with flags directly received from LHC beam instrumentation, and other information provided by Ethernet, to derive energy, intensity, and a set of flags and values representing the LHC machine state. Some of these flags are transmitted directly to the extraction Beam Interlock System (BIS) ensuring the safe transfer of beam between the SPS and LHC machines [1].

Data from both CISG is sent to the arbiter board (CISA) which assembles the redundant information and determines the overall state of the system, before transmitting the information to client systems through the LHC General Machine Timing (GMT) .

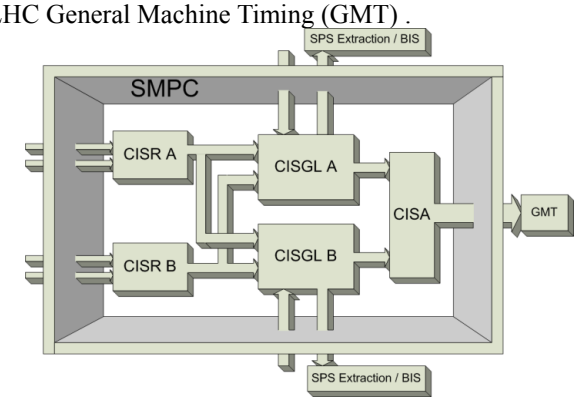


Figure 1: LHC SMPC and clients.

Each board has a pair of Field Programmable Gate Arrays (FPGAs). A "control FPGA" tasked with the execution of critical operations, in which the SMP mission-critical functions are implemented. And a "monitor FPGA" which does not interfere with the critical part, but receives information from it, formats and makes it available for observation and interpretation through the Java software supervision.

Functions

The main goal of the SMPC is to transmit energy and intensity values and to translate input information into states/flags. Principle flags are:

- The Beam Presence Flag (BPF), which evaluates to TRUE when a beam is circulating in the LHC. This is directly received by each CISGL and sent to SPS extraction and CISA.
- The Setup-Beam Flag (SBF), which evaluates to FALSE when the circulating beam is considered as dangerous for the tunnel equipment (magnets, etc) if any problem inducing the loss of the beam occurs.
- The Moveable Devices Allowed In / Stable Beams flags (MDI/STB) are used by the four main experiments (ATLAS, CMS, ALICE and LHC-b). The MDI flag is set to true when a specific energy is reached, when the beam is squeezed at each of the four interaction points, and when the appropriate beam mode is set. Similarly the STB becomes true when the beam mode is stable in addition to the MDI being true.

SAFE MACHINE PARAMETER TESTER

As the SMPC has a heavy dependency on external equipment, it is not possible to test it completely in the real machine environment, it receives information from and transmits information to a diverse set electronics in both the LHC and SPS machines. The SMPT reproduces the machine environment by sending all the different information to the SMPC, replicating the characteristics of each system connected to the SMPC. The SMPT is implemented in a separate VME crate, and a LabVIEW program supervises and controls the SMPT, as well as reading back the states of each element of the SMPC, checking if its behaviour is as expected.

The VME Crate SMPT

The task of the SMPT is to simulate all the inputs (energy, intensity, flags, etc), retrieve the output signals and cross-check the state of the SMPC versus the prescribed function in the specification. The SMPT uses the same hardware boards as the SMPC with different VHDL programs (see Figure 2):

- five CISTR are used to simulate energy and intensity frames (in yellow) sent to the CISRs, and the BPF sent to the back panel of the SMPC (in blue),
- a CISTA is used to read back and verify the output of the CISA which normally goes to the GMT (in blue),
- two CISTCL read the current loop flags directly sent by the back panel from the CISGL to the SPS extraction lines (BPF, SBF) and to the BIS.

On Figure 2, the SMPT VME crate is represented on top with the front panel outputs and inputs which connect to all SMPC connectors on the front panels (on bottom left) and back panel (on bottom right).

Fibres optical connections are represented in yellow, electrical ones in blue, and current loops in green.

LabVIEW Software

The SMPT is monitored and controlled through a LabVIEW program realising the actions required for each test. The front panel of the program, shown on Figure 3, represents the hardware state of each SMPC board, including their connections, and their internal states. A colour-coding is used to display the state of each element:

- dark gray for elements which are not tested (The majority of elements on picture 3),
- dark blue for elements which are currently or are going to be under test (The four SBF equations of CISGLB),
- green for elements successfully tested (the CISA energy and all its dependant tests) and
- red for elements whose test has failed (The intensity priority test in the CISGLA).

Three users' actions are possible:

- "Ctrl click" on an object (rectangle, label or frame) to launch this specific test. On picture 3, the SBF label of CISGLB was clicked to obtain the blue objects,
- "Ctrl shift click" to launch this specific test and all dependant ones. On picture 3, this has been done on the Energy object of the CISA which has induced the test of all green objects
- "Shift click" to show the diagnostic of a test which has been executed, this opens a window showing a breakdown of the test steps.

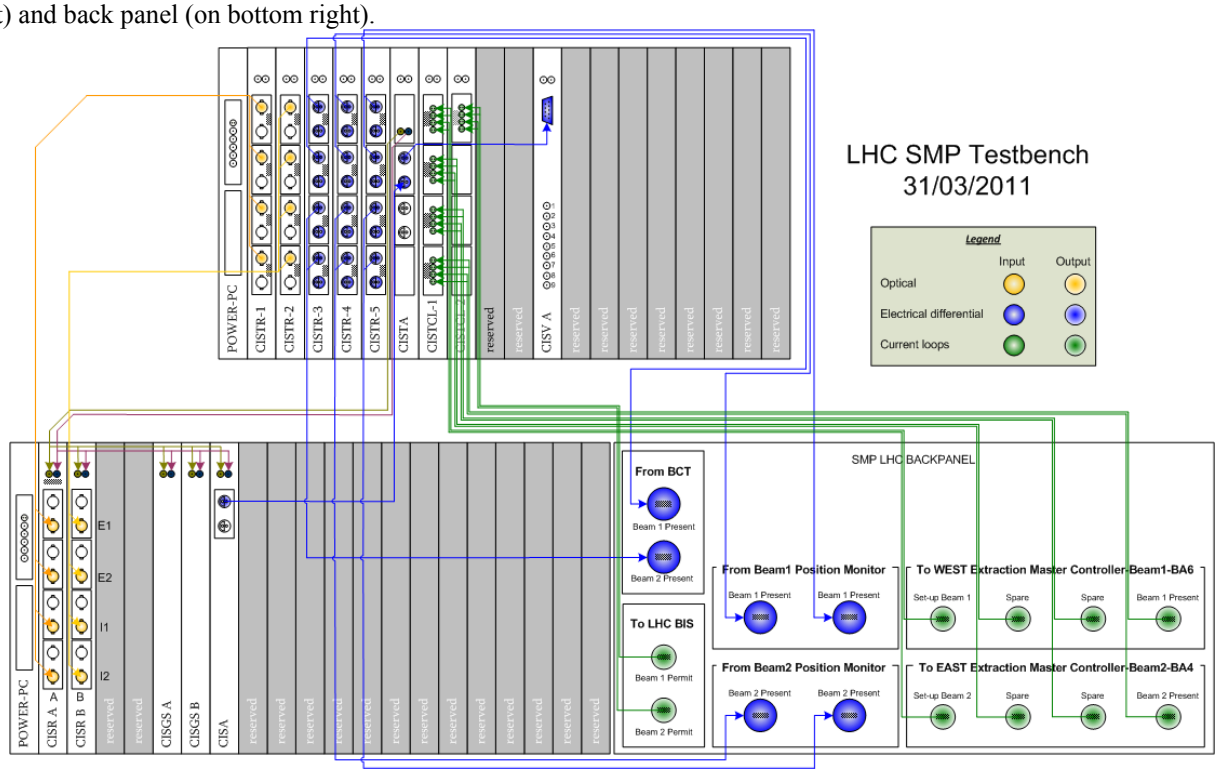


Figure 2: LHC Connexion between SMPT and SMPC.

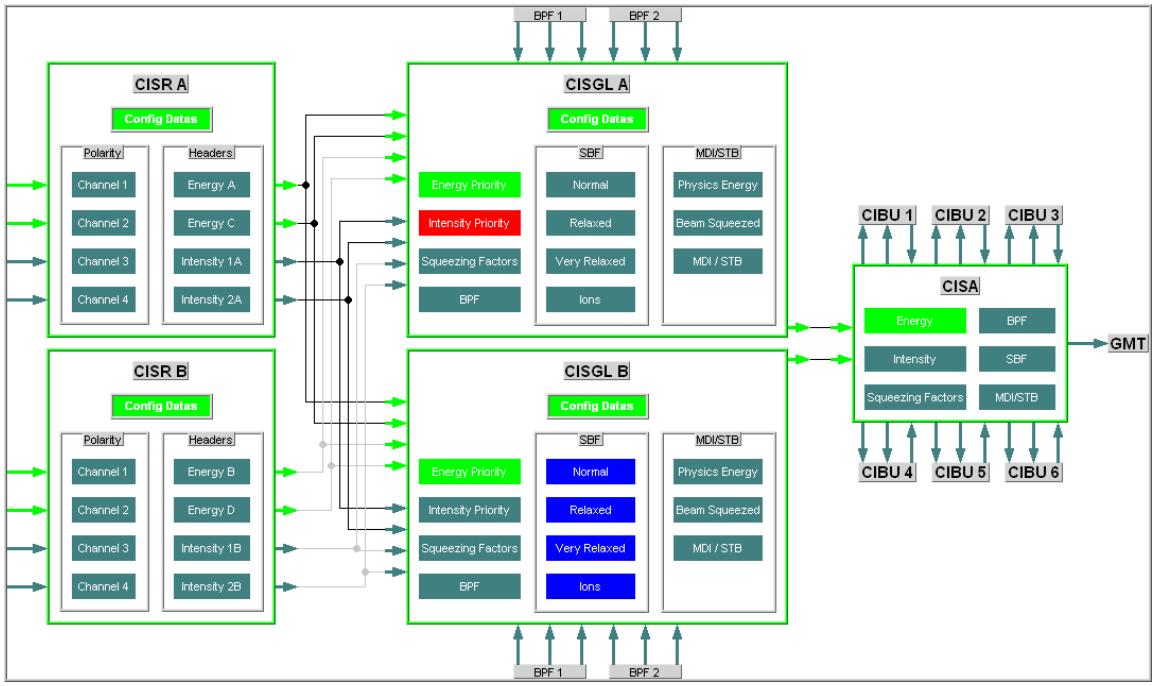


Figure 3: Front panel of the LHC LabVIEW tester.

FESA Layer

The communication between the program and the VME crates SMPC and SMPT is done through a CERN standard architecture called FESA. FESA is a C-based set of drivers developed at CERN to communicate with several devices, and it provides information which can be interpreted by several programming languages such as LabVIEW or Java.

In this application, FESA is used to access all registers and history buffers (HB) implemented in each monitor FPGA. Each HB is a 1024 record rolling buffer which contains specific events with their time of occurance and status, and which are written only when necessary by the mean of triggers inside the SMPC hardware. The registers and the HB allow information about the status, intermediate calculus and timing parameters of the different functionalities to be determined. This information is shown on the front panel of the LabVIEW tester in the picture 2.

The reading is done through simple requests or subscriptions with a refresh rate of one Hertz.

TEST METHODOLOGY

"V" Approach and Predicate Logic

The "V" approach is based on the desire to have the SMPC entirely reproduce to specification requirements. The SMPC was developed from the specification and, in parallel, a set of tests were implemented to validate the behaviour described in the specification, without taking into account how the SMPC was built. As a normal test system, the SMPT allows the validation of the behaviour of the SMPC with respect to the specification. This parallel development has the added advantage of highlighting any kind of ambiguity in the specification itself.

This "V" approach is a good way to consolidate the specification, at the same time, the initial specification was not written with this in mind. To avoid misunderstanding, misinterpretation or un-defined conditions, further effort was carried out to generate a documentation free of ambiguity, using Formal Methods (FM). FM are used to describe the behaviour of a system, independently of its realisation (hardware, software, abstraction...), even if there are several ways to realise a system, depending on the target where the logic is foreseen to be established. FMs in this case took the form of Predicate Logic, similar to a programming language, giving an exhaustive description of how the system has to work, with only a single interpretation.

For example, the SBF is calculated for each particle beam from several input conditions. There are four equations (Normal, relaxed, very relaxed, ions) which give an intensity threshold *THD* (i.e. a number of particles) for a specific LHC proton beam energy. If the LHC beam intensity is below this threshold, the SBF is true (the beam is a set-up beam), otherwise it's false.

For each beam, the SMPC receives two intensity values *IA* and *IB*, each one associated with a "valid intensity" flag, respectively *IA_valid* and *IB_valid*. The intensity value used for the calculation is the highest if both are valid. If not, the maximum intensity value (all bits to '1') is chosen as it automatically sets the SBF to false, which is the fail-safe value (as it is the more dangerous case). The SBF can be expressed as the expression:

$$SBF = (IA\_valid)\&(IB\_valid)\&(IA \leq THD)\&(IB \leq THD)$$

This expression is the basis used to build the SMPC and to define for the SMPT which tests can validate the SMPC behaviour.

Test Structure & Organisation

Full tests coverage is impossible to reach, as 428 bits are derived by hardware and software connections. Moreover, for the LHC SMPC, 82 32-bits registers can be written to change the system configuration and behaviour. This represents a total of 2^{3052} input combinations, without taking timing requirements into consideration.

The main approach to reduce the number of tests is to modify only the inputs which have effect on parameter under test. Moreover, complex functionalities can be tested in steps, by validating intermediate results first.

Tests involving thresholds are a particular case. The threshold and the threshold value plus one are fixed numbers which are always tested. But to be more complete, it's important to add tests of the intervals below and above the threshold. As all values cannot be considered, the solution kept is to choose a random value inside each different interval.

The test bench software has been built to facilitate the addition, modification or suppression of tests. The tests are not defined in the program itself but in an Excel file named "Tests to do" on Figure 4.

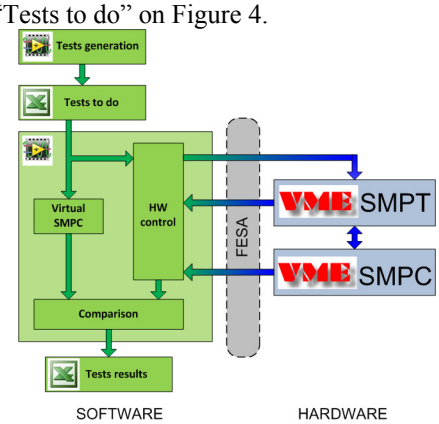


Figure 4: Tests organisation structure.

This file contains each possible elementary test divided in steps. A step is defined as a set of actions which are realised at the same time. Eight actions (all optional) are defined, each one in a separate worksheet:

- three write: for the hardware it can be frames or flags, for the software it is registers,
- four read: for the hardware it can be frames or flags, and for the software registers or HB records,
- one sets the delay before starting the next step.

The writing of this file requires a meticulous approach, so another LabVIEW program (Tests generation on igure 4) has been realised to facilitate the definition of tests.

Tests Sequence

As tests are defined by the program on-the-fly, it's impossible to know in advance what the expected results are. This is also due to some stimuli which have to be chosen randomly at the very start of tests. For this reason, it is mandatory to introduce a subprogram able to calculate at the end of each step the expected internal state (registers and HB) of every board. This "virtual" SMPC is built directly from the specification (as the real SMPC) and gives the possibility to validate the information gathered from the hardware.

The virtual SMPC reads the test steps and transforms them into an array which each cell is a combination of all states of all SMPC inputs for a specific time. The time granularity is 1ms, thus a 5 second test sequence generates an array of 5000 cells. The outputs signals and the FPGA memory are also simulated and calculated for each step, and then sent to the Comparison module.

In parallel, the test steps are read by another subprogram, the hardware control (HW control), which formats them into commands to be executed by the SMPT, waits for the real delays, and reads signals and memories of the SMPC and of the SMPT. All the gathered information is compiled to have a similar structure than the virtual SMPC results, and is also sent to the Comparison module.

The Comparison code compares the real and the virtual data and gives the results following the colour code of Figure 3 (green if all is the same and red otherwise). In case of failed test, it's possible to open the diagnostic window of the program; the results are also saved in an Excel file (Tests Results on Figure 4).

FUTURE PLANS AND CONCLUSION

The main improvement concerns the reduction of time consumption for tests, as well for the SMPC than for the virtual SMPC. In total the SMPC needs more than 5 hours to be tested; the virtual SMPC needs 20 minutes. Presently, the SMPC communicates with the hardware using FESA, and needs to wait for the data to be refreshed before readings. This refresh occurs every second, asynchronously with the test process. Then the time to wait before having new data is between a few milliseconds and a second. Instead of using FESA, LabVIEW could connect on the VME crate through a SSH connection and read the memories thanks to low-level programs which already exist for manual operation. This would allow an improvement in the time for testing the SMPC. Concerning the virtual SMPC, the actual method is to simulate iteratively the memories and outputs at every millisecond, which becomes huge for tests of more than 1 minute. This can be replaced by an equation independent of time, capable of evaluating the system state in a single calculation, given all inputs. In this case any test would complete within a few millisecond. Such an equation has already been defined for the CISR.

To conclude, FM and "V" approach ensure having specification out of ambiguity and a system which behaves as specified. These techniques are particularly efficient as the system is more complex, as for the SMPC version 3.1, currently in use in the CERN control room.

REFERENCES

[1] B. Todd et al. "The architecture, design and realisation of the LHC Beam Interlock System", ICALEPCS'05, Geneva, Switzerland, October 2005.

THE DIAMOND MACHINE PROTECTION SYSTEM

M. T. Heron, S. Lay, Y. Chernousko, P. Hamadyk, N. Rotolo,
Diamond Light Source, Oxfordshire. UK

Abstract

The Diamond Light Source Machine Protection System manages the hazards from high power photon beams and other hazards to ensure equipment protection on the booster synchrotron and storage ring. The system has a shutdown requirement, on a beam mis-steer of under 1msec and has to manage in excess of a thousand interlocks. This is realised using a combination of bespoke hardware and programmable logic controllers.

The structure of the Machine Protection System will be described, together with operational experience and developments to provide post-mortem functionality.

INTRODUCTION

Diamond, a third generation 3GeV synchrotron light source[1], commenced operation in January 2007. The storage ring (SR) is based on a 24-cell double bend achromatic lattice of 561m circumference. It uses a full-energy booster synchrotron and a Linac for injection. The spectral output is optimised for high brightness up to 20keV from Undulators and high flux up to 100keV from Multipole wigglers. The current operational state includes twenty photon beamlines, with a further twelve beamlines now under design or construction.

The design of the Diamond control system [2] is based on the EPICS control system toolkit. It primarily uses VME IOCs as the plant interface. The EPICS control system undertakes plant supervision and operation within defined limits, but does not protect equipment from damage. Independent from the control systems there are hardware and PLC-based subsystems to ensure correct operation of process-based equipment, e.g. cryogenic systems, and protection of equipment. These systems constitute the Machine Protection System.

MACHINE PROTECTION SYSTEM REQUIREMENTS

The Diamond machine protection system is required to protect equipment from damage, both at the local and the global level, by managing the hazard. This is achieved in most cases by controlling the source of the energy. At the global level this involves protecting the vacuum vessel from damage by the photon beam, and series-connected magnets from thermal damage. At the local level this again involves protecting magnets from thermal damage, and photon front ends and vacuum equipment from damage. The MPS further ensures that photon front ends are only operated when insertion devices are within approved ranges of beam current and gap, and that vacuum conditions are such that Gas Bremsstrahlung production is within acceptable levels.

When the required response times for the protection of this equipment are considered, it is evident that most occurrences can be addressed using commodity PLCs. However, the protection of the vacuum vessel against the thermal load of a mis-steered beam requires the dumping of the stored electron beam in under 1 msec (including detection, logic, transmission and beam dump). This therefore requires a dedicated hardware-based solution.

In view of the size of the Diamond facility, a fibre optic infrastructure was installed from each of the accelerator instrumentation areas to a common area. This has been utilised by the MPS, thereby minimising cable runs and potential earth loops.

The MPS is required to ensure protection of equipment, but to operate independently from the supervisory layer of the control system, the EPICS IOCs. The latter ensures correct operation of the equipment within prescribed limits, and only when this fails should the MPS act. Similarly, the Personnel Safety System [3] at Diamond is independent of the Control System and the MPS, and ensures the protection of personnel from various hazards. At the point where the MPS and PSS systems interface, the MPS issues requests to the PSS, with the PSS having the final control of any radiation source.

Whilst the PSS at Diamond is designed against a process based on the standard IEC61508, the MPS design process is not so rigorous, but adopts good practice in managing hazards. It uses simple logic to enable functional testing, solutions designed to fail safe and undergoes periodic validation re-testing to ensure ongoing system integrity.

PRINCIPLE AND ORGANISATION

The MPS monitors a large number of interlock signals from diagnostics instrumentation, vacuum instrumentation, photon front ends and plant monitoring subsystems. Based on logic it can then remove the source of the energy to ensure protection of equipment. Depending on requirements, interlocks are managed on a Local or a Global basis. The Global system is structured as two layers, and supports fast- and slow-response-time interlock requirements.

A Global MPS (GMPS) module takes the interlock permits for a given interlock circuit from each of the cells of the accelerator, and, subject to all interlocks being good, produces a permit to operate the source of energy: the RF amplifier for vessel protection and the PSU for magnet protection (see Figure 1). The Local MPS module takes fast Interlock inputs from one cell of the Storage Ring or one quadrant of the Booster. Fast interlocks are those that must drop the beam in under 400μsec (the maximum speed of the interlock) in the event of failure.

Examples of this type of measurement are signals from electron beam position monitors, beam blow up, beam obstruction (vacuum isolation valves) and reduction in the quality of the vacuum in the storage ring. The module also takes the permits from PLCs that manage the slow interlocks for that local cell.

Slow interlocks are processed by PLCs which manage interlocks where a response time greater than 100msec is acceptable. These include water cooling, vessel and magnet temperature, and component interlocks from photon front ends and photon beamline control subsystems. The PLCs also produce local permits for the correct operation of equipment controlled from within the same cell, an example being the Quadrupole magnet PSUs.

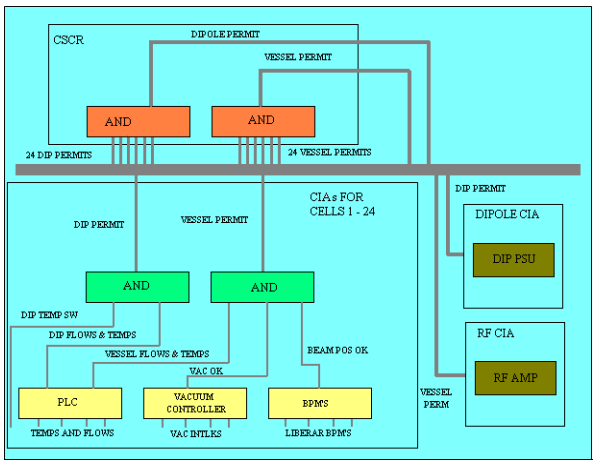


Figure 1: Architecture.

System Realisation

The interlock system is based around DLS-designed machine protection cards. The interlocks feed to the LMPS card from the monitoring equipment, and if all interlocks are good, the enable signal is generated. The enable signals for each cell are fed via fibre links to a GMPS card which produces the permit required by the RF or Dipole Power supply to operate.

The Local MPS Card

Each Local MPS (LMPS) card can receive up to 32 interlocks as input signals and can be configured as one or two independent channels. The LMPS modules are located in a VME-based IOC that provides power, time reference and an interface to the control systems for monitoring. The output permits are produced as a 5MHz pulse train and are distributed over optical fibre links to the GMPS module. For a good permit the 5MHz clock is the active state, so that is possible to differentiate between a failed interlock, which gives zero output from the optical receiver, and a broken fibre, which gives a high output from the optical receiver.

The 32 inputs to the LMPS card are optical-isolated to support 24 volt signal levels. Each input is ORed with a signal from a jumper so that it can be overridden (or disabled) by inserting the jumper. The outputs from all the OR gates, which represents the status of the used inputs, are then ANDed to give an overall logic state. This is then latched on by a Reset signal from the controls systems. Each LMPS module is monitored by a Hytec 8001 digital I/O board to read the state of the jumpers and inputs, to provide reset signals and to monitor the transition to fault condition. The loss of an individual interlock removes the output and sends a signal to the Hytec 8001 board to latch the transition. The output can only make the transition to the enabled state when all the interlocks are good and a reset signal from the 8001 is asserted.

The Global MPS Card

The Global MPS (GMPS) cards form a “star” point for the Local MPS cards. The GMPS provides the same functionality as the LMPS card, but takes 5MHz optical inputs as the interlocks. Each card produces 4 permit outputs as optical signals.

The RF/Dipole Interface Card

The interface from the GMPS to RF amplifiers and dipole power supply is provided by the MPS decode module. The 5 MHz event stream is detected to provide a logic level, which is made available as a relay clean contact, opto coupled and TTL output.

A variant of this decode module provides logic functionality to cascade the Vessel Protection interlock from the RF amplifier to the Dipole power supply. This is to address the possibility that an attempted switch-off of the RF amplifier fails to dump the beam. In this case, the failure to dump the beam is detected, causing the MPS signal to the SR dipole to be removed. To date this variant has not been implemented.

The PLC Subsystem

The PLC subsystem uses OMRON CJ1 & CJ2 PLC hardware [4] integrated into a customised chassis. The decision to encapsulate the PLC into crates was taken to provide an element of future-proofing, as the PLC inside the chassis can be upgraded to keep up with advances in technology, whilst maintaining the same plant interface. It also helps maintain consistency of layout and simplicity of construction of control system cabinets. Using a standard 3U 19” crate for the PLC encapsulation with defined I/O interfaces has further enabled consistent I/O assignment and rapid on-site commissioning. An example of this is shown in Figure 2.

Vessel thermal protection is realised using a PLC crate for monitoring slow interlocks associated with water flow and temperature of the vessel and other components. This can monitor 432 digital inputs and 288 Thermal inputs, and provides 96 local permits, which connect in to the LMPS or direct to local PSUs.

Variations of the PLC subsystem have been realised for thermal protection, for control of vacuum valves, and for ID gap and Gas Bremsstrahlung protection. In all cases they build on the same hardware components, the same

software components and the same interface to the control system, thereby ensuring good standardisation.

Figure 2: PLC sub system for control vacuum valves.

Communication

Communication between EPICS and the Omron PLC's associated with the MPS is realised using the Omron FINS protocol over a standard serial connection. This allows reading from and writing to single registers or block registers, where registers can be data memory or I/O locations (writing to direct I/O was disabled).

Summary of Implementation of MPS on the SR & Booster MPS

The storage ring (SR) MPS is structured as two circuits: vessel protection and dipole magnet protection. For each circuit the GMP module receives 24 permits from LMPS modules, one per cell of the SR. The LMPS modules each receive permits from the local PLC module for that cell. For the vessel permit, there are also interlocks from the photon front-end PLC systems and from the photon beamlines. This is shown in Figure 3.

The booster MPS is structured as four circuits: vessel protection, dipole magnet protection, F-quad protection, and D-quad protection. For each circuit the GMP module receives 4 permits from LMPS modules, one per quadrant of the Booster SR. The Local MPS modules each receive permits from the local PLC module for that cell.

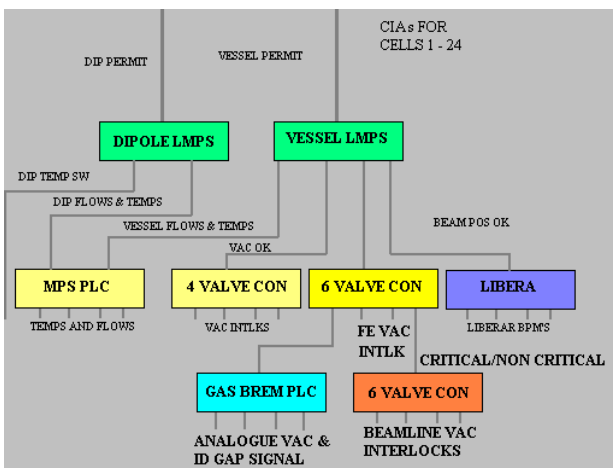


Figure 3: Beamline/Front end connection.

PLC Software

Diamond has structured the PLC ladder logic in such a way as to provide a standard set of drivers for all applications that were deemed likely. The code is structured to ensure that the software engineer, when integrating the unit into a system, has to follow the same pattern regardless of where the system is being used, and what it is being used for. This is to permit easy interpretation and fault finding, and to allow additions to be made to that system by other engineers.

The standard programme structure is broken down into a number of sections for housekeeping (where revision control, hours run and watchdog operation are managed), input conditioning, interlock chains, device drivers, power -supply unit enables and interlock forwarding.

Generally only the interlock logic section and the interlock forwarding need any adjustments to suit the location.

CONTROL SYSTEM INTERFACE.

EPICS provides the user interface to the MPS system. An EPICS driver communicates with the MPS card via a Hytec 8001 64-bit digital I/O board with modified firmware. This allows the reading of the state of the 32 inputs and 32 jumpers directly and previously unused bits in the 8001's control and status register indicate the interlock state and provide the means to generate a reset signal to enable the interlock. The system also allows each permit to be dropped as part of the start-up testing after a shutdown. An overview of the whole system is presented to the operator which shows the status of the global permits. It is then possible to drill into the LMPS via the IOC crate in which it is based. It is further possible to see the status of each individual input on a PLC via the serial link between the PLC and the IOC.

PERFORMANCE

In terms of performance, the most critical requirement is the dumping of the electron beam to protect the vacuum vessel. This has been measured as ~600 µsec and is the sum of:-

- BPM detection delay ~200 µsec,
- Delay between BPM output from Global vessel permit ~250 µsec,
- Delay of GMPS OP through RF amplifier and cavity to beam dump. ~ 150 µsec.

OPERATIONAL EXPERIENCE

The system has proven to be very easy to commission and allows good scope for expansion as upgrades to the storage ring are carried out. By using the modular approach, commissioning could be carried out in isolation on each section as it was completed. This was particularly useful to DLS as the SR was not completed in consecutive sections.

Many of the interlocks for the MPS originate within the CIA, and because all of the field wiring is brought back into the MPS rack, a lot of the initial fault finding could be carried out without the need for vault entry. This saved

time and reduced disruption during the machine commissioning phase.

Occasional problems with spurious tripping that did not give an event record were an issue that has prompted a re-design of the LMPS for the reasons described below.

FUTURE DEVELOPMENTS

Post Mortem Buffer

The arrangement of latching the failed interlock is supposed to capture the first interlock to be lost, thereby diagnosing the original cause of the beam loss. On a beam loss, the beam position interlocks activated within the clock window are also latched along with the true first interlock. If the first interlock is a non-position interlock, it will come up along with all the beam position interlocks and so can be identified. Also, if the first interlock is a position interlock, the post mortem in the Libera BPMs provides clear information as to the cause. However, this falls down in the event of a spurious interlock from a BPM, or from a LMPS to the GMPS module. The latter caused a series of false beam dumps due to inadequate transmission power between the LMPS and the GMPS module and reduction of optical output with aging of transmitters, so that spurious failures of interlocks were registered by the GMPS. The originating LMPS module was only identified by monitoring the decoded inputs to the GMPS with a logic analyser. This has resulted in a redesign of the GMPS to add post-mortem capability, and improved transmitter power on LMPS modules to ensure an adequate optical budget.

Remote IO PLC

For the next phase of photon beamlines a new generation of Omron PLCs has been selected. These have enabled the adoption of Remote IO and an Ethernet interface between the control system and the PLC.

Remote I/O provides a way of reducing cabling complexity and improving flexibility on deploying I/O into the field. RIO uses standard Ethernet cabling and Profinet, and RIO modules are built up into standard junction boxes, one for thermal monitoring and another for general purpose I/O. The Remote I/O does not perform any logic on any of the signals; it simply acts as a receiver and passes everything back to the PLC crate. This necessitates configuration of RIO output modules such that in the event of failure of RIO communication I/O fails to the safe state. The PLC logic is protected by a watchdog timer to protect against failures in communication with RIO.

The new PLC CPU (CJ2M) has a built-in Ethernet connection which allows both EPICS and the programming software to share the connection. EPICS communication is handled over UDP using FINS protocol whilst programming is realised over TCP/IP.

CONCLUSION

In summary the DLS Machine Protection system is modular in structure which has facilitated deployment and maintainability. It has met its design specification for shutdown time and has proven its worth on a number of occasions when there have been vacuum incidents or localised increases in heat due to cooling problems.

REFERENCES

[1] R. P. Walker, “Overview of the Status of the Diamond Project”, EPAC 2006, Edinburgh

[2] M. T. Heron et el, “The Diamond Light Source Control System”, EPAC 2006

[3] M Wilson “Development of The Diamond Light Source PSS In Conformance With EN 61508” ICALEPCS 2011, Grenoble

[4] http://www.omron-industrial.com/uk/home/products/automationsystems/ProgrammableLog/ModularPLCSeri/CJ1M/default.asp

PERSONNEL PROTECTION, EQUIPMENT PROTECTION AND FAST INTERLOCK SYSTEMS: THREE DIFFERENT TECHNOLOGIES TO PROVIDE PROTECTION AT THREE DIFFERENT LEVELS

D. Fernández-Carreiras, D. Beltran*, J. Klora, J. Moldes, O. Matilla, R. Montaño, M. Niegowski, R. Ranz*, A. Rubio, S. Rubio-Manrique, CELLS, Cerdanyola del Vallès, Barcelona, Spain

Abstract

Alba [1] is a synchrotron light source under installation located nearby Barcelona. This 3 GeV third generation light source is planned to deliver the first X-rays beam to the users in 2012. The Linac has been commissioned in 2008, the booster in 2010 and the Storage Ring in 2011. The seven Beamlines included in the "Phase One" are being commissioned at the end of 2011.

The Fast Interlock System, Equipment Protection System (EPS) and the Personnel Safety System (PSS) ensure that the operations are done safely for the machine components and people.

All three use independent hardware and communication channels, and they guarantee different response times and safety levels. The Fast Interlock System, works in the microsecond range, the EPS in the millisecond, and the PSS has a cycle time of 150 milliseconds. However, the PSS has some extra requirements, since it is related to human lives safety, and therefore it requires the highest possible reliability. It ensures a Safety Integrity Level 3 (SIL3) according to the international norm IEC 61508.

FAST INTERLOCK

The Fast Interlock is implemented upgrading the single channel links channels employed by the Timing system, to a bidirectional system. It is based on events, relying on a tree architecture where the root (event generator) gathers and redistributes the interlock events from/to all the leaves (event receivers). Each event receiver is configured to perform an action (i.e. activate pulse on an output) for a particular event code. The cards (event generators, event receivers and fiber optics fan-outs) have been produced by MRF [2]. There is one event generator and about 88 Event Receivers distributed, which currently have configured about 480 connections to equipments [3].

When an interlock signal is produced in the Beam Position Monitors (BPM), Radio Frequency plants (RF), and Front Ends (FE), it is transmitted back to an event-receiver adjacent to the event generator which redistribute the events to the whole tree. The fiber optic links have all a fixed length, 200 meters, required for ensuring the precision in the synchronization events. The time between the generation of one interlock event in one node, and the reception is in the order of four microseconds.

The fast interlock system, provides accurate timestamps of each event (interlock), allowing a 8 nanosecond resolution in the discrimination of interlock-events for postmortem analysis.

*On leave

EPS

Besides few hundreds of interlock signals, which require an action in the microsecond range, there are other seven thousand managed by the EPS. The EPS guarantees a transmission in the millisecond range. Typically the cycle times are below the 20 ms. The EPS is responsible for all interlocks in the machine and the beamlines. All signals managed by the Fast interlock are also backed up by the EPS.

The Equipment Protection System manages permits and interlocks avoiding damaging the hardware. It is built on B&R PLCs [4] with CPUs installed in cabinets in the service area and distributed I/O modules installed in shielded boxes inside the tunnel and in the beamline hutches. CPUs and remote periphery are interconnected by the X2X fieldbus. A deterministic network, Ethernet-PowerLink, interconnects all CPUs to each other.

Architecture and Technology

The EPS is built on a network of PLCs and remote peripheries. The EPS for the Accelerators includes 56 B&R CPUs X20CP1484, all equipped with Ethernet interface 100 Base-T and Ethernet PowerLink.

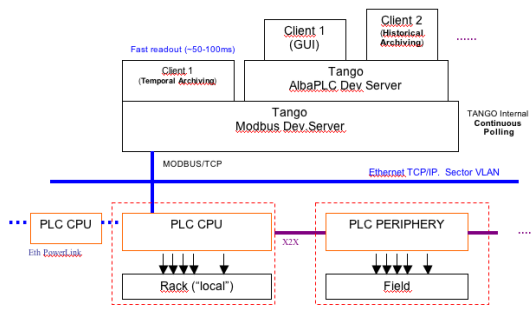


Figure 1: Architecture of the EPS PLC System.

In addition, 110 main periphery cabinets are linked to the CPUs by the X2X bus. Lead-shielded boxes are installed inside the Tunnel to make cables shorter, cheaper and easier to install. The deterministic Ethernet PowerLink network runs on a separated industrial switch. It is a deterministic OSI Level2 network. Interlocks between different subsystems are transmitted over this media. Every Beamline has an independent system, usually composed by one CPU and 2 main remote peripheries, having few hundreds of signals. Figure 1 shows the conceptual design of the EPS.

The standard Ethernet network handles communication with Human Machine Interfaces, Archiving services, etc.

The PLCs is connected to the main control system VLAN in that sector (or beamline). A Tango device server polls values from PLC CPUs using Modbus/TCP. On top of this, another Tango device Server, AlbaPLC, provides dynamic attributes named according to the field devices, in order to make easier the integration in the SCADA.

Automatic Code Generation

The controls and cabling database stores the information concerning not only cables, but equipments, channels, connectors, domain names, boot servers, etc. In particular all equipment codes with channels names and connections are stored in this database.

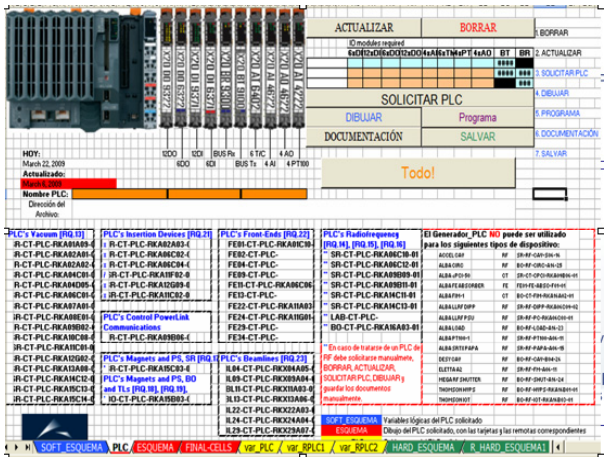


Figure 2: Excel application for EPS code generation.

Most of the code in the PLC is generated automatically from the cabling database. A Visual Basic Script running in excel generates another file containing the declaration of variables, data structures, software tasks, modbus mapping and documentation. Figure 2 shows a view of this excel file. The parts of the code still excluded from the automatic generation are the logic conditions. Additionally, Tango attribute names and the expert GUIs are also generated from the controls equipment and cabling database.

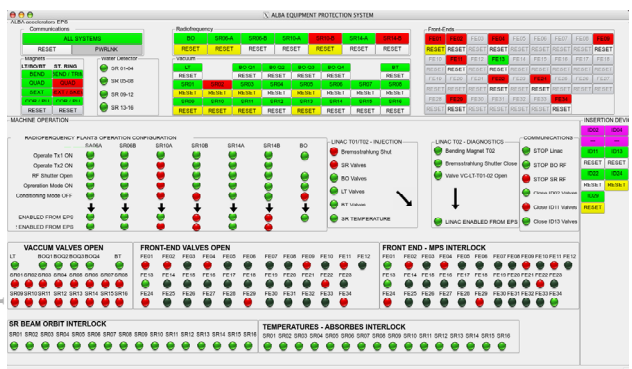


Figure 3: The main EPS GUI for the Machine.

Subsystems and Logic

The EPS covers mainly six subsystems. Those are: Vacuum, Magnets, Radio Frequency, Insertion devices, and Front-Ends. Besides, every Beamline has an independent EPS system, connected to the Accelerators with eight wired signals. The logic of each subsystem is complex and adapted to every particular case. As a general rule, when a problem arises, any other subsystem shall be informed. In particular if a vacuum valve closes as a result of an overpressure in a vacuum pipe, the RF shall be shut down and the beam killed. All these communications are transmitted over PowerLink, although there are some cases considered more critical which are also wired from one PLC to another, like interconnections of different vacuum sectors or as we have just mentioned, the links between Beamlines and Front-Ends. Figure 3 shows the main user interface for the Equipment Protection System.

PSS

The PSS is an independent system built on Pilz Safety Programmable Logic Controllers (PLCs) [5]. It manages access to restricted areas, such as Linac, Tunnel and Beamline lead hutches, and surveys radiation levels. It prevents people to get a radiation dose higher than the limits given by the law. It is subjected to Ionizing Radiation Regulations. It is independent from any other system in Alba and the Spanish Nuclear Safety Council shall approve it.

The PSS is governed by the international norm IEC 61508. It rules the use of electrical and software driven safety systems to provide risk reduction up to an acceptable level. The ALBA PSS Safety Integrity Level aims SIL3. Besides this, the PSS follows the rule of "Redundancy and Diversity". Redundancy will be achieved by having two independent lines for every signal. Diversity means that any action will be applied to two different parts of the system, for example disabling the RF comprises revoking the permit for the RF driver and for the High Voltage Power Supplies (HVPS). In other words, each action results in two redundant outputs. The installation of the system has been outsourced and is being achieved by the companies Pilz and PROCON. Although the PSS is one independent system, It has two parts: Accelerators y Beamlines, The accelerators PSS controls the Tunnel and The Linac bunker. It consists of two PLCs intercommunicated by the safety bus. The tunnel has four access doors, whereas the linac bunker has one door.

Hard X-Ray beamlines have two hutches controlled by the PSS. In the case of Soft X-Rays beamlines only the Optics hutch is lead-shielded and controlled by the PSS. A beamline has a "Independent" PSS with a dedicated PLC connected to the Acelerators PSS (Main) by the Safety Bus. It can be disconnected from the Main PSS, only in some well defined safe conditions. In case of failure it provokes the Main PSS goes to safe state.

Functionality

There are 24 radiation monitors, manufactured by Thermo [6] (FHT 6020A controller) are distributed in the service area and experimental hall. They monitor both

gamma and neutron doses integrated over four hours and providing two alarm levels. Each alarm level has a Pilz PNOZ S4 Safety Relay, activated when the radiation reaches the alarm threshold. All inputs/outputs to/from the PSS are digital. Every door has two different limit switches, one of which works also as magnetic lock (Pilz PSEN lock, and PSEN code).

Interlocking the doors once the restricted areas have been evacuated ensures access control. In order to make sure that a zone is clear of personnel, a search patrol is needed. A Search patrol is started from the control room and performed in bunker and tunnel by two authorized people. Once the Permit is given in the control room, the patrol begins. Nobody but the persons performing the patrol are allowed in the bunker and tunnel. Search buttons all around the tunnel are pressed in sequence. Every button has to be pressed in a time interval, not before a minimum time and having a timeout. The PSS has four main states, OPEN (free access), INTERLOKED (once the search patrol has been completed, RESTRICTED, and SAFE. An indicator shows also the presence of beam in the restricted areas. The restricted access function is implemented only for the Accelerators PSS in order to make short interventions in the restricted areas without needing a new patrol. This restricted access mode is allowed in the control room to one (up to six) authorized person (magnetic card). After taking one of the personal keys this person can go in the restricted area. Once all keys are in place in both the door cabinet and the control room, the system goes to "interlocked" again. When an unsafe condition arises, the system goes to safe state. Doors are unlocked only after a decay time, unless an emergency stop is pressed.

The Control Room

The SCADA and the Operation keys are in a cabinet in the control room. The SCADA, only meant for monitoring and diagnostics, reads tags from standard (not-safety) data blocks in the PLCs. It does not have write access to the PLC. The operation permits are given with physical keys. The Pulsed power supplies and the Booster and Storage Ring bending magnets can also be manually disabled from the control room.

Architecture and Technology

The system is built around the Pilz PSS SB2 3006-3 ETH-2 CPU, two of them for the accelerators, tunnel and linac, and one for each beamline. All CPUs are intercommunicated by a Pilz Safety Bus (certified SIL3). Keys, door switches, emergency stops and relays are also SIL3. Safety relays are installed in radiation monitors, Bending magnet power supplies, Radiofrequency (RF) High Voltage Power Supplies (HVPS) and Inductive Output Tubes (IOT), as well as in other electronic boxes like the RF detectors and the Electron beam current detectors. The RF waveguides and Front-End Shutters have also SIL3 PILZ PSEN 1.1p20 switches.

Logic

The logic is organized in permits and interlocks, a permit is the result of a set of conditions fulfilled. An interlock is a condition for granting or revoking permits. For example, in order to open the front-end, the hutch must be interlocked, emergency-stops armed, the operation keys in place, etc.

CONCLUSION

Nowadays, the PLC technology is used in a wider range of applications that traditionally were linked to other technology. Safety PLCs are today a common choice in the industry for high risk environments where a failure might have many people killed, like trains, etc. Also, standard PLCs are cheaper, smaller and more powerful and we found a large variety in the market today. A distributed system combining Ethernet (used as a fieldbus) and a proprietary X2X fieldbus is proven to be cost-effective solution. Periphery can be closer to the devices, making cabling easier and cheaper. Where the required response times are several microseconds, a solution with PLCs is not viable anymore. Those cases are often a reduced subset and can be accomplished with "ad-hoc" solutions. For this particular case, the upgrade of the Timing system, to support bidirectional links for implementing fast Interlocks, took place. It was proposed by Alba and Implemented and made available in the market by MRF.

CONTRIBUTIONS

Many people has worked in these projects, in particular we would like to thank X. Queralt, the Safety Officer, S. Marchal, B. Saló, J. Jamroz (electronics group). Also, special thanks to P. Berkvens, the safety officer of the ESRF, for his help and his advises with the PSS logic. The EPS is one of the most important customers of the cabling database, developed and maintained by CELLS Management Information System group, especially I. Costa and O. Sánchez.

REFERENCES

[1] CELLS-ALBA. http://www.cells.es
[2] MicroResearch Finland. http://www.mrf.fi
[3] O. Matilla et Al, "The Alba Timing System. A known architecture with a Fast Interlock System Upgrade". These proceedings.
[4] B&R Automation. http://www.br-automation.com
[5] Pilz. http://www.pilz.com
[6] http://www.thermofisher.com

ARCHITECTURE FOR INTERLOCK SYSTEMS: RELIABILITY ANALYSIS WITH REGARD TO SAFETY AND AVAILABILITY

S. Wagner, A. Apollonio, R. Schmidt, M. Zerlauth, CERN, Geneva, Switzerland
A.Vergara-Fernandez, ITER Organization, St.Paul-lez-Durance, France

Abstract

In the design of interlock loops for the signal exchange in machine protection systems, the choice of the hardware architecture impacts on machine safety and availability. The reliable performance of a machine stop (leaving the machine in a safe state) in case of an emergency, is an inherent requirement. The constraints in terms of machine availability on the other hand may differ from one facility to another. Spurious machine stops, lowering machine availability, may to a certain extent be tolerated in facilities where they do not cause undue equipment wearout. In order to compare various interlock loop architectures in terms of safety and availability, the occurrence frequencies of related scenarios have been calculated in a reliability analysis, using a generic analytical model. This paper presents the results and illustrates the potential of the analysis method for supporting the choice of interlock system architectures.

INTRODUCTION

For particle accelerators like the LHC and other large experimental physics facilities like ITER, the machine protection relies on complex interlock systems. They are required to trigger machine stops in case of emergency, but not spuriously. *Machine stop* implies, for example, the extraction of the energy stored in magnet powering circuits and, in case of the LHC, the extraction of the beams from the machine.

For the interlock loops protecting the LHC superconducting magnet circuits, spurious triggers of machine stops, lowering availability, can be tolerated to a certain extent since they do not affect the longevity of the equipment. In ITER's case on the other hand, high machine availability, and therefore limited spurious triggers of machine stops are required since each fast stop causes significant magnet aging due to the induced forces. The number of tolerated spurious machine stops for the LHC lies in the range of a few tens per year (around 10% of all fills), while for ITER it is expected to be limited to only a few in the whole lifetime of 20 years .

In conjunction with the development of a prototype for ITER interlock loops, a reliability analysis comparing six possible interlock loop architectures has been performed.

This paper in the first part introduces the method and the generic model used for the analysis. The second part discusses some results of the performed studies. The third part introduces the approach developed for the verification of the model and intermediate verification results.

METHOD

The following sections summarise the most relevant aspects of the interlock loop model used for the analysis. It is based on the method introduced in a study of a part of the LHC Machine Protection System [1].

Interlock Loop Model

The model reflects an interlock loop ('system') with 4 components, as illustrated in Fig. 1.

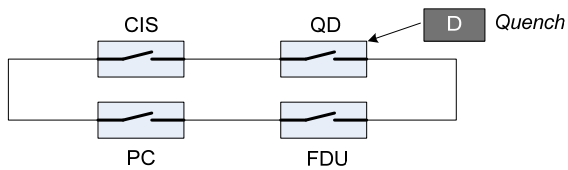


Figure 1: Basic model of interlock loop with 4 components.

The components are considered to be switches, which can fail in two modes, *blind* and *false*, according to failure rates λb and λf (Fig. 2, left). The system demand is modelled by virtual component *D,* following demand rate x (Fig. 2, right).

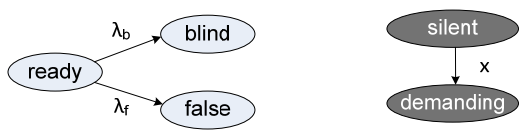


Figure 2: State diagrams reflecting component behaviour and system demand.

The different states represent the following conditions:

- *Ready*: switch closed, ready to open upon demand (initial state)
- *Blind*: failed closed, not ready to open upon demand
- *False*: Switch open, spuriously or upon detection of switch-internal failure (i.e. without demand)
- *Silent*: nominal condition of monitored machine equipment, no demand (initial state)
- *Demanding*: Emergency condition detected, demanding machine stop (loop opening)

The possible state configurations occurring within a given observation time tf, represent four scenarios. They are exemplified by means of a generic quench loop, including components QD (Quench Detection), FDU (Fast Discharge Unit), PC (Power Converter) and CIS (Central Interlock System, Fig. 1). In case a quench of a magnet is detected (i.e. system demand), the QD is triggered to open the loop, thus informing the PC to switch off the power supply and the FDU to extract the energy from the magnet powering circuit:

(1) *Mission completed*: neither quench nor spurious loop opening during an operational cycle
(2) *False trigger* (→Preventive stop): loop opening due to failure of any switch (mode *false*), without quench
(3) *Demand success* (→Emergency stop): loop opening by QD upon quench
(4) *Demand missed* (→Missed emergency stop): missed loop opening upon quench due to QD failed closed (mode *blind*)

Scenario 4 (worst case scenario) includes the potential of severe damage to the machine, hence interfering with machine safety. Together with scenarios 2 and 3, it defines the machine availability reflected by scenario 1.

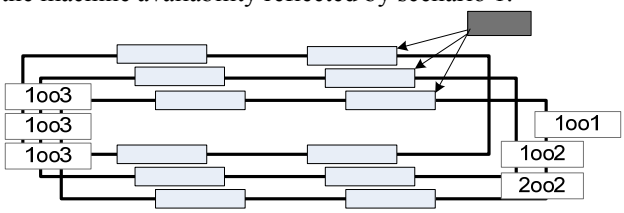


Figure 3: Models of interlock loop for different architectures (featuring up to three redundant lines).

Figure 3 gives an overview on the models for the six architectures under consideration, derived from the generic model introduced above:

- 1oo1: single-line solution, no redundancy
- 1oo2: two redundant lines with 1-out-of-2 logic
- 2oo2: two redundant lines with 2-out-of-2 logic
- 1oo3: three redundant lines with 1-out-of-3 logic
- 2oo3: three redundant lines with 2-out-of-3 voting
- 3oo3: three redundant lines with 3-out-of-3 logic

For a *False trigger* of *2oo2* for example, both lines need to be open due to switch failures.

Model Input Parameters

Table 1 summarises the model input parameters, representing seven 'degrees of freedom' for case studies.

Table 1: Model Input Parameters

Parameter	Description
Components:	
λf	Failure rate *false*
λb	Failure rate *blind*
x	Demand rate
Operation:	
tf	Observation time
Architecture:	
k	Number of lines
n	Number of components/line
-	'Voting'

Model Assumptions

The presented model includes a series of assumptions and simplifications:

- Independent failures of components
- Identical components (with regard to reliability, i.e., failure rate)
- All components in initial state at t=0 (system 'as-good-as-new')

Besides these rather common assumptions, there are a few others more specific to an interlock loops:

- Any switch opening is recognised as 'line opening', independent of the (potentially *blind*) state of components in the same line. This reflects a loop with redundant readout.
- The demand is limited to one loop component (i.e. QD). The demand signal is fault free and, in case of redundant lines, is simultaneously distributed to all lines (Fig. 3). This neglects possible redundancy of the triggering source.
- A switch opening due to failure mode *false* is permanent, i.e. an open switch stays open till the end of current operational cycle. This neglects transient failures.
- The voting (included in the 2oo3 architecture) is fault free. This assumes a technical solution for the voting that does not add significant complexity that might lower reliability.

Analytical Model Description

The model uses an analytical description [1,2] adapted and extended to the characteristics of the system under consideration.

The most relevant improvement concerns the inclusion of voting and the (related) elimination of the *False missed* scenario (being absorbed in the remaining four scenarios).

Model Implementation

The analytical model description is implemented using Maple like in the previous studies [1,2].

RESULTS

Table 2 shows the default values of the input parameters, defined as the starting point for extended case studies.

Table 2: Default Input Parameters

Parameter	Default value		Comment
Components:			
λf	1E-4	[h^{-1}]	MTTF: 12 months
λb	1E-5	[h^{-1}]	MTTF: 15 years
x	2E-4	[h^{-1}]	MTTF: 6 months
Operation:			
tf	720	[h]	30 days
Architecture:			
n	4	[-]	cp. basic model

The values for the component failure rates (λf, λb) and the demand rate (x) are derived from MTTF estimations based on experience with the LHC. The observation time (tf) of 30 days reflects the expected length of an ITER

Table 3: Scenario Probabilities for the Different Architectures (Default Input Parameters)

	1oo1	1oo2	2oo2	1oo3	2oo3	3oo3
Mission completed	6.50E-01	4.88E-01	8.12E-01	3.66E-01	7.31E-01	8.52E-01
False trigger	2.33E-01	4.10E-01	5.68E-02	5.43E-01	1.42E-01	1.40E-02
Demand success	1.17E-01	1.03E-01	1.30E-01	9.08E-02	1.27E-01	1.32E-01
Demand missed	3.99E-04	1.54E-06	7.97E-04	6.57E-09	4.60E-06	1.19E-03

operational cycle, i.e. the continuous operation between two maintenance periods.

The number of components per line (n) corresponds to the basic model (Fig. 1 and 3). The remaining architecture parameters are defined by the architectures under consideration.

Table 3 presents the results for the case study based on the default input parameters:

- With regard to *Demand missed*, the 1oo3 architecture is top (lowest probability), while with regard to *Mission completed* the 3oo3 architecture is (highest probability).
- With regard to the combined aspects, the 2oo3 architecture solely ranks in the 'top three solutions' for both *Demand missed* and *Mission completed*.
- Compared to the 1oo1 architecture (ranking second best with regard to the combined aspects), the 2oo3 architecture features a decrease of *Demand missed* and an increase of *Mission completed*.

Translating these observations in terms of safety and availability, the following statements result (for the given default parameters):

1. With regard to safety, the 1oo3 architecture is top, while with regard to availability it is the 3oo3 architecture.
2. With regard to the combined aspects, the 2oo3 architecture is a best-compromise solution, ranking top three for both safety and availability.
3. Compared to the 1oo1 architecture, the 2oo3 architecture includes an increase of both safety and availability.

In order to prove these statements, a series of sensitivity analyses have been performed. In the following, the results of the variation of failure rate *false* (λf) are presented. The further sensitivity analyses based on the variation of the remaining input parameters (*blind* rate, demand rate, observation time and the number of components per line) are beyond the scope of this paper.

Variation of λf between 1E-7 h^{-1} and 1E-2 h^{-1}

The variation reveals that while statement 1 holds for the entire considered range of λf , statements 2 and 3 are true (T) for λf ≤ 1E-4 h^{-1} only, not for higher λf (Table 4).

The reason is that for higher λf, the 2oo3 architecture exceeds the 1oo1 architecture in terms of *False trigger* (Fig. 4, crossing line), which results in a decrease of *Mission completed* compared to the 1oo1 architecture (Fig. 5). Hence, as of a certain parameter range, the 2oo3 architecture is outperformed by 1oo1 in terms of availability.

Table 4: Assessment for Statements 1 to 3 for λf Varied between 1E-7 h^{-1} And 1E-2 h^{-1}

Stat.	1E-7	1E-6	1E-5	1E-4	1E-3	1E-2
1	T	T	T	T	T	T
2	T	T	T	T		
3	T	T	T	T		

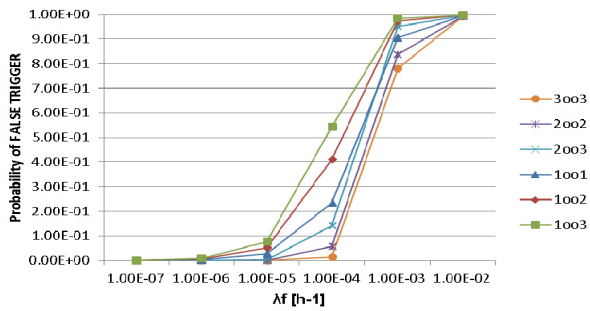


Figure 4: Probability of *False trigger* against λf between 1E-7 h^{-1} and 1E-2 h^{-1}.

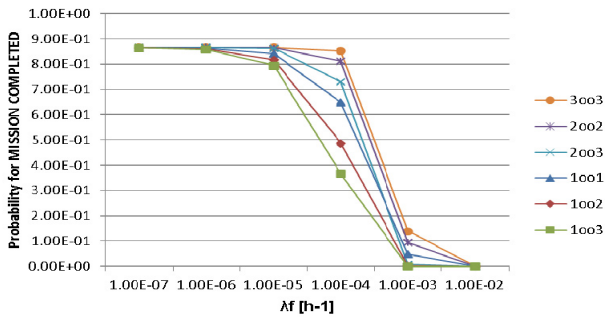


Figure 5: Probability of *Mission completed* against λf between 1E-7 h^{-1} and 1E-2 h^{-1}.

Summarising the above observations, the following extension to the three statements is to be made:

4. Statement 2 and 3 require reasonably low failure rates *false*, λf .

The confirmation of statement 1 is provided by Fig. 5 and 6, showing the advantage of the 3oo3 architecture with regard to availability (Fig. 5) and the advantage of the 1oo3 architecture with regard to safety (Fig. 6).

Discussion

The lower performance of the 2oo3 compared to the 1oo1 architecture for high λf can be explained by the total number of components included in the architectures. As of a certain λf, the voting is no longer able to compensate

for the (three times) higher amount of components. However, such high failure rates are not reflecting electronic devices in use nowadays. Their rates are expected to be in a lower range.

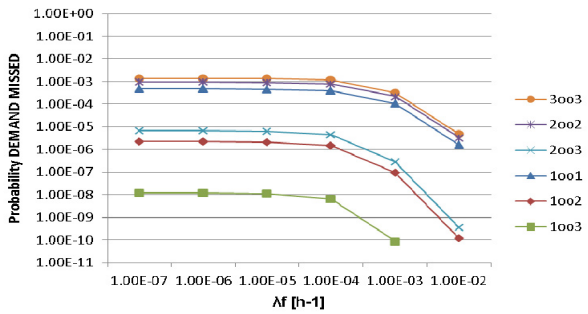


Figure 6: Probability of *Demand missed* against λf between 1E-7 h^{-1} and 1E-2 h^{-1} (logarithmic scale).

The constraint expressed by statement 4 only relates to the availability aspect. The 2oo3 architecture does appear in the top three with regard to safety for the entire considered range of λf, hence outperforming 1oo1 in terms of safety.

The introduced method allows for additional comparison from a different point of view. Instead of assessing the performance of the architectures based on given input parameters, the architectures can be compared in terms of the reliability of the components required to achieve a desired system performance. This may lead to statements like *in order to achieve an availability X, the 2oo3 architecture allows for Y orders of magnitude higher component failure rates compared to other architectures*.

The presented analysis has been taken into account in the decision-making with regard to the design of a prototype for the ITER quench loops. Considering the results of the analysis and the possibility for testing and maintenance during operation provided by redundant architectures (which can counter the related disadvantage of increased amount of components), a 2oo3 solution is being envisaged.

VERIFICATION

As mentioned above, the analytical model description used in this analysis is a further development of the approach introduced in earlier studies. For the verification of the results, and the underlying analytical description, a study based on Monte-Carlo simulation has been started.

The simulation of a single operational cycle includes two basic steps:

- Generation of random numbers representing the times of state transitions (i.e. of component failures or system demand), according to the input parameters (failure and demand rates)
- Assignment of the resulting time sequence of state transitions (within the given observation time) to one of the four scenarios

The simulation of a multitude of operational cycles then allows for statistical analysis:

- Derivation of the relative occurrence frequencies of the different scenarios

The simulations are implemented using Matlab. Two independent approaches are being developed which differ in the scenario assignment step:

- Explicit assignment based on the time sequence of random numbers
- Implicit assignment based on a graphical model representation including step functions and signal transmission (implemented using Simulink)

The intermediate results of the explicit approach show good agreement with the results presented in Table 3, indicating relative errors in the range between 1E-6 and 1E-2 for the frequent scenarios (based on 8E7 simulated cycles). For the rare scenario *Demand missed*, the error is not meaningful since the number of simulated cycles is too low for a reasonable accuracy.

The implicit approach is still under development. Currently, there are results available on the 1oo1 architecture only and with fewer simulated cycles due to a significantly increased need of simulation time compared to the explicit approach. The intermediate results show a relative error in the range between 1E-4 and 1E-1 for the frequent scenarios (based on 5E4 simulated cycles).

CONCLUSIONS

This paper presents the method and results of a reliability analysis addressing the properties of various interlock loop architectures with regard to machine safety and availability. It shows the advantages of a 2oo3 architecture for systems with high requirements in both safety and availability.

Further application and development of the method is ongoing. Subsequent studies are being performed addressing the interface between interlock loops (e.g. quench loop) and the protected machine subsystems (e.g. magnet powering circuits), including detectors.

The Monte Carlo approach for the verification of the different studies is being further developed. In addition, the validation of the models is to be addressed, in particular with regard of the fault-free voting assumption underlying the model.

REFERENCES

[1] Wagner, S. (2010), "LHC Machine Protection System: Method for Balancing Machine Safety and Beam Availability", Doctoral Thesis (Diss. ETH No.19043), ETH Zurich

[2] Wagner, S., et al. (2009), "Reliability Analysis of the LHC Machine Protection System: Analytical Description", in Particle Accelerator Conference 2009 (PAC09), Vancouver, BC, Canada.

SECURING A CONTROL SYSTEM: EXPERIENCES FROM ISO 27001 IMPLEMENTATION*

V. Vuppala, J. Vincent, J. Kusler, K. Davidson, NSCL, East Lansing, MI 48824, USA.
vuppala,vincent,kusler,davidson@nscl.msu.edu

Abstract

Recent incidents of breaches, in control systems in specific and information systems in general, have emphasized the importance of security and operational continuity in achieving the quality objectives of an organization, and the safety of its personnel and infrastructure. However, security and disaster recovery are either completely ignored or given a low priority during the design and development of an accelerator control system, the underlying technologies, and the overlaid applications. This leads to an operational facility that is easy to breach, and difficult to recover. Retrofitting security into a control system becomes much more difficult during operations.

In this paper we describe our experiences with implementing ISO/IEC 27001 Standard for information security at the Electronics Department of the National Superconducting Cyclotron Laboratory (NSCL) located on the campus of Michigan State University (MSU). We describe our risk assessment methodology, the identified risks, the selected controls, their implementation, and our documentation structure. We also report the current status of the project. We conclude with the challenges faced and the lessons learnt.

INTRODUCTION

NSCL's distributed control system uses Experimental Physics and Industrial Control System (EPICS), and is managed by the Electronics Department (EE). While attempting to secure the control system, it became evident that it could not be done in piecemeal fashion. Hardening one part of the system does not suffice; the weaker links in the chain are either obscured or ignored, and leave the entire system as vulnerable as before. So EE wanted to address security in a holistic manner, and decided to implement the ISO/IEC 27001 Standard for information security.

Confidentiality, Integrity, Availability

The cornerstones, basic principles, or foundations of information security are Confidentiality, Integrity, and Availability (CIA). Confidentiality ensures that only authorized personnel have access to information. Integrity ensures that the information remains valid by guarding against unauthorized modifications and destruction. Availability guarantees that the information is available whenever requested (by authorized personnel).

* This work was supported in part by the National Science Foundation under the Cooperative Agreement PHY-06-06007.

ISO/IEC 27000 STANADARDS

ISO/IEC Standard 27001 and 27002 form the crux of the 27000 series of standards. ISO Standard 27001 (based on British Standard 7799 Part 2) provides guidance to establish, implement, operate, review, and improve an Information Security Management System (ISMS). ISO 27002 (based on British Standard 7799 Part 1) describes the best practices to manage information security risks. ISO 27001 presents a management system: a framework of policies, procedures, guidelines and associated resources to achieve the security objectives of the organization. ISO 27002 presents a set of controls: means to manage security risks.

ISO 27001 advocates an iterative process-based approach built on Plan-Do-Check-Act (PDCA) model to establish and manage an ISMS [1]. It recommends four phases for ISMS: establish, implement and operate, monitor and review, and maintain and improve. It mandates management responsibilities, internal audits, reviews, and continuous improvement of the ISMS.

ISO 27002 is divided into eleven clauses [2]. Each clause is divided into categories. Each category has an objective and a set of controls to achieve that objective. The security clauses in ISO 27002 are:

- Security Policy
- Information Security Organization
- Asset Management
- HR Security
- Physical Security
- Communication and Operations Management
- Access Control
- Information Systems Acquisition, Development, and Maintenance
- Information Security Incident Management
- Business Continuity Management
- Compliance

An organization is certified against ISO 27001 and not ISO 27002. Annex A of ISO 27001 refers to the controls of ISO 27002.

ARGUS THE ISMS

In this section we describe the implementation of Argus, our ISMS. Its roadmap is shown in **Figure 1**. We first defined Argus' scope (it was limited to the EE department and related support services), and the guiding policy for the ISMS. Next we chose the OCTAVE Allegro [3] as our risk assessment methodology. Using this approach we identified our critical information assets:

information that is important to us. This included controls and PLC software, documentation of our systems, software licenses, EPICS archiver database, and IOC configurations. Then, we identified the containers of the information assets. The containers can be of three categories: technical (server, software, hardware etc), physical (paper, folders etc), and human (intellectual property, ideas etc).

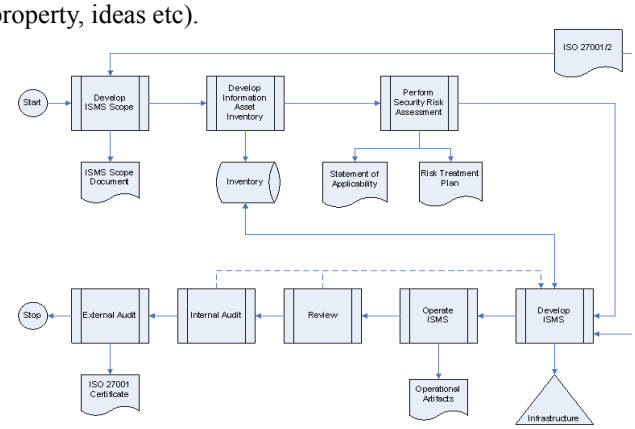


Figure 1: Argus Roadmap.

Risk Assessment

As the next step, we identified the conditions (areas of concern) that can affect the information assets or their containers. Then we qualified them with actors, means, and outcomes. This resulted in a list of threats to our assets. An example of such threat is: anyone with access to control network can modify the IOC configuration files. We then evaluated the impact (see Table 2) of every threat based on a set of measurement criteria (

Table 1). This gave us a relative risk score (RRS) for each risk. An example of this score is shown in Table 3 for the risk - "inadvertent modification of EPICS channel values".

Table 1: Risk Measurement Criteria

Impact Area (IA)	IA Priority
Safety and Health	5
Reputation	4
Financial	3
Legal	2
Productivity	1

Table 2: Impact Values

Impact	Value
No Impact	0
Low	1
Medium	2
High	3

Table 3: Relative Risk Score for a Risk

Impact Area (IA)	IA Priority	Impact Value	Score
Safety and Health	5	Low (1)	5
Reputation	4	Med (2)	8
Financial	3	High (3)	9
Legal	2	None (0)	0
Productivity	1	Low (1)	1
Relative Risk Score			23

We used the RRS to prioritize the risks. . Based on the relative risk score and probability of risk occurrence, we could categorize the risks into various levels. An example of such risk level matrix is shown in Table 4.

Table 4: Risk Levels

Probability	Relative Risk Score			
	60+	40 to 59	20 to 39	0 to 19
High	Level I	Level I	Level II	Level III
Medium	Level I	Level II	Level II	Level IV
Low	Level II	Level II	Level III	Level IV

We treated the risks based on the risk level or priority. Risk treatment involved one of the following actions:

- Avoid the risk by using the controls from ISO 27002 or controls developed in-house
- Reduce the risk by using the controls
- Accept the risk or residual risk. If the risks are of low probability and incur high cost for mitigation, they may be accepted. However, all acceptable risks must be documented and approved by EE department head.
- It is possible to transfer a risk by insuring against it but we did not have any such risks.
- It is also possible to share risks, with vendors of other labs, but we did not encounter such risks.

Documentation

Documentation forms a critical part of any management system. The policies, procedures, standards, and guidelines related to Argus are structured in a hierarchical fashion. Policies refer to related procedures which in turn point to relevant guidelines, standards etc (see Figure 2). The top level policies and procedures are linked together in the *Argus Security Handbook*. The existing document system used by the lab for ISO 9001, 18001, and 14001 management systems is also being utilized for Argus' documentation.

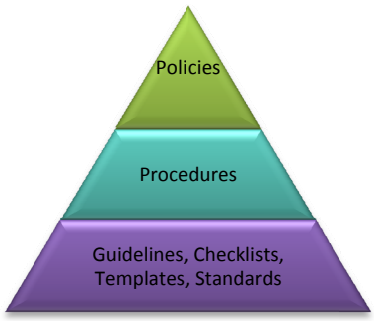


Figure 2: Argus Documentation.

		Information Class				
		Class I	Class II	Class III	Class IV	Class V
Access Medium	Control Network	Not Allowed	No Controls for PVs and Embedded Controllers. Authorization for other data.	Authorization, Encryption	Authorization, Encryption	No controls for read. Authorization, encryption for write.
	DAQ Network	Not Allowed	No controls for read. Authorization for write.	Authorization, Encryption	Authorization, Encryption	No controls for read. Authorization, encryption for write.
	Office Network					

Figure 3: Access Control Matrix.

ARGUS CONTROLS

The controls used in Argus are identified in Argus Statement of Applicability. A security policy for the department and a policy for its periodic review were defined. An organization structure is put in place to focus on information security and management of Argus. It consists of Information Security Board, IT Group, and Information Security Manager. The EE department head leads this organization. The physical and human resource security policies and procedures are based on current practices which were found to be adequate. For disaster recovery, the backup tapes are taken off-site on a weekly basis. A mechanism for live off-site backups, to a remote facility, is being implemented. Business continuity plans and procedures are defined but not tested. Many of the standard operations and communications management controls were found to be adequately covered by the current practices; the rest were implemented.

The current Trouble Reporting System used for the existing ISO based management systems (9001, 18001, and 14001) is being utilized for security incident management. Legal, statutory, and contractual compliance policies are based on NSCL's and MSU's policies.

The existing software development and project management policies and procedures were incorporated into Argus. New policies and guidelines on secure software development practices were developed.

Access Control

The information assets were classified into five categories, Class I through V, Class I being the most sensitive and Class V being least sensitive. Information assets in the department can be accessed through various means: Internet, Michigan State University Wired or Wireless Network, NSCL Controls Network, NSCL Office Network etc. Access controls were defined based on the information class and access method. An example access control matrix is shown in Figure 3. To improve security, the various networks within the lab were segregated, through a firewall, at the end of last year.

ARGUS LIFECYCLE

Argus is a living system; it continuously improves itself through reviews, audits, and feedbacks. The phases and activities of its lifecycle, defined in *Argus ISMS Policy* and *Argus ISMS Procedure*, are summarized below.

- Plan
  - Define Scope and ISMS Policy
  - Develop Approach to Identify, Evaluate, and Treat Risks
  - Identify and Analyze Risks
  - Evaluate Risk Treatment Options
  - Select Controls to Treat Risks (Statement of Applicability)
- Do
  - Develop Risk Treatment Plan (RTP)
  - Implement RTP
  - Measure Effectiveness of Controls
  - Manage Information Security Incidents
  - Implement Training and Awareness Programs
- Check
  - Monitor and Review Argus
  - Conduct Internal Audits
  - Measure Argus' Effectiveness Based on Audits, Incidents, Feedback etc
  - Review Risk Assessment
- Act
  - Identify Improvements Based on Reviews/Audits
  - Identify and Implement Corrective and Preventive Actions

RETROSPECTION

Challenges

Control Systems have been designed, by vendors and the community, with little emphasis on security. They are not designed to guard against malicious code or unauthorized access. So they have to be secured through external means such as management procedures, user training, and network isolation. It is also difficult to harden control system platforms such as PLCs. We found that it is difficult to implement secure software development processes. Programmers should understand and guard against security issues like buffer overflows,

memory leaks, SQL-injection etc which add to their programming effort. Static and dynamic source code analysis tools are useful but require programmers to learn to use them.

The educational and research oriented environment in the lab is also not favourable for implementing security procedures. Changing the culture of the organization is a challenge. Security conflicts with convenience, and finding the balance is difficult.

Lessons Learnt

ISO 27001 is an extensive standard; implementing it is an onerous task. However, it is not necessary to implement it across the entire organization in one shot. The standard allows the scope to be adjusted. Hence, it is crucial to start small, implement it, and then expand. For the initial iteration, start with the current practices, document them, establish the initial ISMS, and them improve upon it. Do not make drastic changes to the current processes; this will only infuriate the users. Remember, users are an important, if not the most important, part of the overall security system.

The most important factor for the successful implementation of any management system, especially ISO 27001, is management support. Without it, the required changes to the organization's culture are impossible.

Leverage the infrastructure of existing management system like ISO 9001. There are several similarities among these standards, which allow the infrastructure and processes to be shared.

ISO 27001 implementation requires support from every unit of the organization, so involve all the units in the process especially during risk assessment. We made a deliberate decision not to use consultants to help us with the implementation. An in-house team is required to manage the ISMS. The consultant may help with the templates and guidance, however bulk of the work still needs to be done by the in-house team. It is worthwhile to train the in-house team in ISO 27001 audit and related trainings.

ARGUS PROJECT

Risk assessment, Statement of Applicability, Risk Treatment Plan, and initial set of documentation have been completed. The registrar for external audit has been selected through a formal bidding process. The external audit of Argus comprises of pre-assessment, Stage I audit, and Stage II audit. The Argus documents are currently being vetted. Internal audit and pre-assessment are expected to be completed by end of 2011.

The project to develop Argus was started in August of 2009; it is expected to finish in the early 2012. The estimated effort was approximately 1000 person hours, of which approximately 800 have been currently spent.

CONCLUSION

. Was the implementation worth the effort and cost? We think so. Due to this exercise, we have a very good insight into our vulnerabilities, threats, and risks. This experience has helped us incorporate security as a design element in the development of our systems. The original intent was to eventually expand this to the rest of the lab. Even though this implementation was not a requirement from our current customers, we feel that it will eventually attract more security-sensitive projects to the lab.

REFERENCES

[1] ISO/IEC 27001 International Standard, Information Technology – Security Techniques – Information security management systems – Requirements

[2] ISO/IEC 27002/17799 International Standard, Information Technology – Security Techniques – Code of practice for information security management

[3] The OCTAVE Allegro Guidebook V1.0, Computer Emergency Response Team (CERT) Program, Software Engineering Institute, Carnegie Mellon University

[4] ISO 27001 Toolkit, http://www.iso27k.com

ACCESS SAFETY SYSTEMS – NEW CONCEPTS FROM THE LHC EXPERIENCE

T. Ladzinski, Ch. Delamare, S. di Luca, T. Hakulinen, L. Hammouti, F. Havart, J-F. Juget, P. Ninin, R. Nunes, T. Riesco, E. Sanchez-Corral Mena, F. Valentini, CERN, Geneva, Switzerland

Abstract

The LHC Access Safety System has introduced a number of new concepts into the domain of personnel protection at CERN. These can be grouped into several categories: organisational, architectural and concerning the end-user experience. By anchoring the project on the solid foundations of the IEC 61508/61511 methodology, the CERN team and its contractors managed to design, develop, test and commission on time a SIL3 safety system. The system uses a successful combination of the latest Siemens redundant safety programmable logic controllers with a traditional relay logic hardwired loop. The external envelope barriers used in the LHC include personnel and material access devices, which are interlocked door-booths introducing increased automation of individual access control, thus removing the strain from the operators. These devices ensure the inviolability of the controlled zones by users not holding the required credentials. To this end they are equipped with personnel presence detectors and the access control includes a state of the art biometry check. Building on the LHC experience, new projects targeting the refurbishment of the existing access safety infrastructure in the injector chain have started. This paper summarises the new concepts introduced in the LHC access control and safety systems, discusses the return of experience and outlines the main guiding principles for the renewal stage of the personnel protection systems in the LHC injector chain in a homogeneous manner.

INTRODUCTION

The access safety system is a vital component of every accelerator facility without which beams cannot be injected and accelerated in a machine. Its principal duty is to ensure that if there is beam in the machine no human being is inside, and if there is a human inside that no beam can be injected.

The Large Hadron Collider access system was put in operation in 2008 following one year of gradual commissioning. It introduced new concepts and safety levels, which were not present in the earlier accelerators at CERN. Within the activities of the LHC injector chain upgrade, the injector complex access safety systems are also being overhauled. Our team is currently involved in the renovation of the Proton Synchrotron (PS) personnel safety system and plans are underway to start the upgrade of the access and safety system of the Super Proton Synchrotron (SPS). These activities should bring the level of safety of the corresponding access systems at least to the level of the LHC system and provide the so much sought after harmonisation of equipment and user experience over the entire accelerator complex of CERN.

SAFETY PROJECT ASPECTS

An access safety system is a complex interlock mechanism acquiring the status of, and acting on, hundreds of Elements Important for Safety (EIS) [1]. We distinguish between EIS-access and EIS-beam. The EIS-access consist of the personnel and material access devices, doors, moveable shielding walls etc. The EIS-beam are accelerator components that can stop the circulation and the injection of beams. The choice of EIS-beam allows redundancy for each interlock chain with technological diversity (e.g. a bending magnet and a moving stopper obstructing the beam aperture). The number of individual components under the responsibility of various organisational units with different approaches to safety provides additional challenges to the overall safety system design and project coordination activities.

In order to achieve the desired level of safety, the safety systems at CERN are designed using the IEC61508 [2] family of standards as a methodology framework. The IEC61508 uses a probabilistic approach to quantify the risks and to check that a system can cope with the requirements defined for each safety function. To this end it introduces the notion of Safety Integrity Level (SIL), which is a measure of safety. It allows to determine the target level of risk reduction that a safety instrumented system should provide. It is scaled from 1 to 4. The higher the occurrence rate of a hazardous event or the severity of its consequences, the higher the SIL level and the implementation constraints. In order to deal with the functional safety aspects, a project strategy has to take into consideration the following aspects [3]:

- preliminary risk analysis;
- specification of the safety instrumented functions with their corresponding SIL level, e.g. stopping the beam in case of an intrusion has been evaluated as a SIL3 function;
- preliminary safety study based on the first version of the functional analysis of the architecture;
- design and implementation of the system based on V-shaped lifecycle model;
- verification and validation of the system;
- organisation of operation and maintenance;
- definitive safety study of the "as built" system, verifying that the SIL of each safety instrumented function has been achieved.

A vital organizational aspect of a safety project is the independence of teams conducting various project steps. This is often achieved by outsourcing the development tasks to external companies. The CERN team participates in the specification and verification phases, the actual implementation is done by a specialized contractor meeting the tender process requirements. The final validation that the system fulfils its mission is done by yet another independent body – the Departmental Safety Officer of the Beams Department conducts an independent test before permitting any beam operation. In addition, throughout the project lifecycle independent consultants are hired to conduct the safety studies and evaluate the SIL level achieved.

THE LHC ACCESS SAFETY SYSTEM ARCHITECTURE PRINCIPLES

Control and Safety Separation

Following the principle of strict separation of the functional safety part from the process control part, the LHC access system is made up of the LHC Access Control System (LACS) and the LHC Access Safety System (LASS) [4].

The role of the LACS is to provide a physical barrier enclosing the LHC accelerator and dividing it into clearly delimited sectors, and to identify the person and verify his or her access authorisations. The LACS controls the access equipment and provides audio and video links between the control room and the field.

The LASS is an interlock system ensuring that no beam can circulate or be injected in case of access operation and that every intrusion detected during the beam operation leads to an immediate stop of the accelerator in a controlled manner. Its EIS-access comprise 40 personnel and 29 material access devices, 203 doors dividing the underground areas into 82 sectors, 17 mobile shielding walls etc. The EIS-beam which can, in parallel to the LHC beam dump system, stop any circulating beams and any injection of new beams are: the mobile beam dumps, horizontal dipole chains and injection septa in the two transfer lines from the SPS to the LHC, the separation magnets in the two collimator regions of the LHC and two reinforced vacuum valves.

The division into two distinct systems allows the use of different hardware, software and testing solutions, each more suited for the specific needs of the subsystem.

Two Channel System

The LASS control system has a distributed architecture and uses the Siemens 417FH Programmable Logic Controllers (PLC). At each of the LHC points a local controller monitors the state of all the EIS of that point and calculates the site resultants which are transmitted to the global controller. The global controller acquires the information obtained from each of the local units and takes the necessary safety actions, calculating a global resultant to enable or disable the global access safety veto. The controller units are linked via a dedicated redundant fibre network routed in the LHC tunnel.

This architecture has been complemented with a relay logic cable loop that provides a technologically redundant logic mechanism to stop the beams in case of an intrusion via the external envelope of the accelerator. In this way, not only the sensors and actuators are provided with sufficient redundancy, but also the central system logic. This goes beyond the commonly used architectures, where a SIL3 system is composed of redundant sensors and actuators and a SIL3 certified logic controller (which is internally redundant providing two execution channels for the safety program). However, the total response time of a system based solely on the PLCs might exceed the needs of the process, as a distributed architecture relies on numerous timeouts before driving the system into a failsafe state in case of a safe failure. Moreover, the IEC61513 standard, a nuclear sector extension of the IEC61508, recommends diversity of means to achieve the safety objectives and thus to minimise a common cause of failure. This is where the simple relay logic steps in.

In the case of the LHC, the relay logic was added to protect the risk of intrusion. For the PS complex this concept has further been expanded to provide redundant logic mechanism that acts also in case of a momentarily lost safe state of an EIS-beam. Contrary to the LHC, which in fact is one accelerator, the PS is divided into several machines with transfer lines providing beam from one to another. In case of a problem with an EIS-beam in one machine, the interlock system immediately acts on the EIS-beams of the upstream accelerator.

External Envelope Inviolability

In a long shutdown period anyone authorised can enter the accelerator. Once this period is finished, all the machine interlocked areas are patrolled to make sure that nobody was left behind. As this process is long, the machines are subdivided into access sectors, each having a binary memory called “search”. The search is armed at the end of a patrol and disarmed only in case of intrusion or an entry in the shutdown period. During beam operation period, short technical stops are often necessary to allow accesses for corrective maintenance interventions. The technicians entering the interlocked zones are not protected by the collective search/patrol mechanism, but instead are given personal protection tokens. As long as all the tokens are not restored, the interlock system will not allow beam operation. This safety concept functions correctly provided there is no possibility of entering the interlocked areas without a token. Hence, the EIS-access of the accelerator external envelope must form an inviolable barrier. In the pre-LHC machines this was achieved by means of video surveillance by the control room operators. The sheer size of the LHC and the number of concerned personnel entering the interlocked areas has put the efficiency of such a solution in question. The external envelope has therefore been equipped with access devices or door-booths, which operate on the same principle as an air-lock

chamber. They are small volumes closed at each end with doors, of which only one can open at a time.

In addition to its volume, which is only large enough to accommodate one person, the Personnel Access Device (PAD) is equipped with a complex automatic system based on ground pressure sensors, infrared radar and photo-electric cells surveying the interior at each passage to eliminate piggybacking and tailgating. Furthermore, the usage of a PAD enhances security making it impossible to trespass the interlocked area. Iris biometric recognition system inside each PAD verifies a match between the access badge used and the identity of the access requester. The access control system, in turn, verifies the person's access authorisations and their validity, and checks the status of the periodic obligatory safety trainings and tests.

The Material Access Device (MAD) is also a door-booth, but of much bigger volume, and allows the introduction of bulky material into the interlocked zones. Each MAD is equipped with a human presence detection system. It comprises infrared barriers close to the doors, two volumetric consumer-off-the-shelf detectors and a motion detection system with a millimetre resolution developed at CERN. The later uses a high resolution digital camera (3M pixels) capable of covering the whole MAD volume (approx. 20m3) and a custom algorithm. By analyzing the digital video frames in real time it estimates precisely the quantity of motion inside the MAD on the basis of small luminosity variations in the frame's pixels. A special analysis of any mutated pixels allows discriminating the real movement from the background noise inherent to all digital images.

LHC EXPERIENCE & NEW CONCEPTS

Usability

The LHC access system was built to a higher degree of safety than its predecessors, but the overall ergonomics has not seen major changes with the exception of the introduction of the door-booths. With the explanations on the use of the devices becoming part of the mandatory safety training, the usability problems have decreased and only sporadically users are rejected e.g. because their backpack obstructs the photo-electric cells in the PAD.

Work Acceptance Tool

Passing through a door-booth requires more time than using a door and this fact combined with the extremely high usage rates during the technical stops (see Table 1) may lead to occasional congestion at the access points.

Table 1: LHC Access Statistics from a 5 Day Long Technical Stop (29.08.2011 – 2.09.2011)

Area	Entry & Exit Passages	Refused Passages	**Total**
Service Area	5'831	243	6'074
Tunnel Area	1'766	13	1'779
Experimental Area	3'209	55	3'264
Total	10'806	311	**11'117**

Analysis has shown that the major congestion factor was the relatively long time it took for the operator to verify if an entry request was in relation to a planned maintenance activity. This was largely resolved with the introduction of the Work Acceptance Tool (WAT), linking an intervention planning tool and the access control system. During technical stops, the WAT automatically limits access to planned maintenance interventions only.

Separation of Token Distribution from PAD Cycle

The LHC access system seldom uses the shutdown access mode ("general" in the LHC terminology), as the day-time maintenance is often interweaved with night-time testing activities requiring the interlocked areas to be patrolled and free of personnel. Hence, in most cases the person entering is in possession of a safety token to preserve the sector search. The delivery of a safety token is integrated with the PAD entry cycle and thus a new token cannot be delivered until the previous person has successfully entered. In the PS, the two actions will be decoupled, with the user first taking the token under the supervision of the operator and then entering the PAD, while the operator can already treat another request.

Maintainability

Maintenance has always been an issue for the access systems as they are required 24 hours a day, 7 days a week; when the accelerator is operating with beam and even more so, when it is in access mode.

Maintenance Doors

Traditionally, the accelerators at CERN had been operated in annual cycles of 7 to 8 months of beam followed by 4 to 5 months of maintenance shutdown. In LHC the long superconductive magnet warming and cooling time has led to a change in the operation calendar strategy with the introduction of short annual shutdown periods and a long shutdown once every few years when the magnets are being repaired. The annual maintenance periods have thus become much shorter making it very difficult to perform the necessary maintenance and verifications activities, especially of the complex access devices of the external envelope. This has resulted in ongoing studies with the goal of moving the external envelope to a second line of protection (e.g. the ventilation doors behind the access devices) during beam operation, thus providing the maintenance teams the time to do preventive interventions on all surface access points while the accelerator is in beam operation.

Removing the EIS-beam from an Interlock Chain

The EIS-beam are surveyed by the LASS permanently. Should they quit their safe state in access mode, the LASS blocks access and, in case of multiple failures, orders evacuation of the interlocked areas. EIS-beam can only undergo maintenance during a complete shutdown of the accelerator complex. This is regulated by a strict procedure. In order to facilitate their disconnection from the system using special "out-of-chain" keys, additional safety functions have been recently introduced. As long as all the EIS-beam are not connected, the upstream chain interlock will not allow beam operation.

Extensibility

An extension of an existing safety system requires the same rigorous approach as the original development project. This strictness filters non-justified *ad hoc* demands and preserves the safety integrity of the system. Until now one major extension has been identified for the LHC access system. It concerns extending the scope of the system not only to cover the radiation hazards, but also risks related to a major helium release in the tunnel and an upgrade is planned for the first long shutdown.

New SIMBA/SIMIT Test Platform

Any change, whether an addition of one EIS or a big extension requires a big testing effort. The LASS test platform [5] provides a test-bed for 2 out of 9 LHC sites at a time. The drawbacks of this test platform are the need for hardware reconfiguration of the I/O modules when changing the simulated LHC sites and a very basic simulator user interface. In the PS, composed of 19 different machines, each with specific configuration, a more versatile test platform is needed to be able to cope with testing of possible extensions. It will be based on Siemens SIMBA module which allows emulation of any I/O configuration without costly hardware reconfiguration and SIMIT software tool facilitating simulation scenarios.

Reliability and Availability

During the past three years of LHC operation, the access interlock has only once caused a spurious beam dump and has been available all the time. The LASS has reacted correctly on all occasions and the safety of the personnel has never been compromised. However, the availability of the LHC for physics has been slightly affected by several spurious sector search drops in access mode, resulting in lengthy patrols. There is no single explanation to the origin of the lost patrols, but they can be attributed to one of the following origins:

- electromagnetic compatibility (EMC) issues;
- lack of synchronisation between control and safety;
- data exchanges between the access point controllers.

EMC Improvements

In the LHC, most of the originally installed magnetic door sensors have been recently replaced by more robust electromechanical contacts. These are not affected by the magnetic fields, but need delicate adjustments. For the PS, a thorough campaign of EMC measurements has been done prior to choosing the access equipment locations.

PAD Control and Safety Synchronisation

In the rest position the LHC PAD inner doors are open. Hence, a safety action applied in the middle of a PAD entrance cycle may in some cases result in the LASS briefly registering both the inner and outer doors opened, which results in a patrol drop. The separation of process control and safety does not preclude synchronisation of the control tasks with the safety actions and the currently designed PS access devices should have one PLC running the two tasks in two processes, with safety having a higher priority, but the control being well synchronised.

Simplified Access Point Control Architecture

An LHC access point composed of one PAD and a MAD is equipped with a total of 5 industrial controllers and a PC which adds to the above mentioned synchronisation issues all the problems of communication between the tasks running in the different controllers. The PS personnel safety system architecture will use only 2 controllers and one PC, as the architecture chosen supports process level integration and not integration of many consumer-off-the-shelf solutions.

Anti-Fraud Detection as a Safety Function

The critical detection of a fraudulent passage in access devices - human presence in a MAD and multiple persons in a PAD - is currently performed by several local controllers. This may lead to the unavailability of the devices in case of failure of one of the controllers. To improve the dependability of these processes it was decided to implement them as new safety instrumented functions of the PS safety system. Moreover, the PS PAD model chosen provides less internal volume making it virtually impossible to fraud.

CONCLUSIONS

The LHC Access System has been in production since early January 2008. The past four years have shown that it has met the desired safety integrity level, thus confirming both the project organisation and the design choices.

In this paper the IEC61508 based methodology used in the project lifecycle phase was presented as well as the important technical principles of the LHC access system. The experience gathered during the operation and maintenance phase was further discussed focusing mostly on improving the system in order to reduce accelerator downtime resulting from access related issues. New concepts have been identified and discussed. They currently start being introduced in the access safety systems of the LHC and its injector chain.

REFERENCES

[1] International Standard IEC 61513 “Nuclear Power plants – Instrumentation and control for systems important to safety”, IEC, 2001-03. The term EIS originates from the French *Elément Important pour la Sûreté* used in this standard. Its English equivalent - Item Important to Safety - is not in use at CERN

[2] International Standard IEC61508 “Functional safety of electrical / electronic / programmable electronic safety-related systems”, IEC, 1998, 2000

[3] P. Ninin “IEC61508 Experience for the Development of the LHC Functional Safety Systems and Future Perspective”, ICALEPCS’09, Kobe, October 2009

[4] T. Ladzinski *et al.*, “The LHC Access System”, ICALEPCS’09, Kobe, October 2009

[5] F. Valentini *et al.*, “Safety Testing for LHC Access System”, EPAC’08, Genoa, June 2008

THE LASER MEGAJOULE FACILITY PERSONNEL SECURITY AND SAFETY INTERLOCKS

Jean-Claude Chapuis, Jean-Paul Arnoul, Alain Hurst,
Mathieu Manson, CEA/CESTA, Le Barp 33114 France

Abstract

The French CEA (Commissariat à l'Énergie Atomique) is currently building the LMJ (Laser Mégajoule), at the CEA Laboratory CESTA near Bordeaux. The LMJ is designed to deliver about 1.4 MJ of 0.35 µm light to targets for high energy density physics experiments. Such an installation entails specific hazards related to the presence of intense laser beams, and high voltage power laser amplifiers. Furthermore, the thermonuclear fusion reactions induced by the experiment also produce different radiations and neutrons burst, and also activate various materials in the chamber environment. All these hazards could be lethal. The SSP (Personnel Safety System) was designed to prevent accidents and protect personnel working in the LMJ.

DESIGN METHODOLOGY

Saftey Studies

For each type of hazard generated by the LMJ process (laser, high voltage, radiations), scenarios of accidents are identified and qualified in terms of gravity and frequency.

Table 1: Gravity v.s. Frequency

Gravity →	Lethal	Impor-tant	Impor-tant	Major	Major
	Major injury	Signi-ficant	Signi-ficant	Impor-tant	Major
	Minor injury	Minor	Signi-ficant	Signi-ficant	Impor-tant
	Incon-fort	Minor	Minor	Minor	Signi-ficant
		Rarely (< 2 times / year)	Some-times (< 20 times / year)	Often (< 200 times / year)	Perma-nent (> 200 times / year)
		Frequency of exposure →			

The table 1 is then used to determine the risk level (i.e. the importance of the potential accident) that is used in the table 2 that indicates the number ant the type of the associated protection barriers necessary to mitigate the risk at an acceptable level.

The CEA security methodological guide defines 2 types of barriers:

- The technical barriers (TB), that are any technical device used to protect the workers, such as access control or safety interlocks,
- The procedural barriers (PB) that involve a human action, that are used in complement of the TB to increase the protection level when necessary.

It also specifies the number of required barriers versus the identified risk level, with 2 options: desirable or acceptable. The choice between those options is generally technical, but it is often also cost driven.

Table 2: Risk Level v.s. Number of Barriers

Risk Level	**Number of barriers**	
	Desirable	Acceptable
Major	3 TB's	2 TB's + 1 PB
Important	2 TB's	1 TB + 1 PB
Significant	1 TB	2 PB's
Minor	2 PB's	1 PB

Functional Analysis

The objective of the Personnel Safety System is to prevent transitions from a safe state to a forbidden state**.**
The safe states are:

- Presence of hazard requiring the absence of personnel,
- Presence of hazard AND all the persons are qualified to work in presence of the hazard,
- Absence of hazard.

The forbidden states are:

- Presence of hazard requiring the absence of personnel AND presence of personnel,
- Presence of hazard AND a person is not qualified to work in presence of this hazard.

So the transitions that must be prevented with adapted barriers are the following:

- Entrance of a person when a hazard requires the absence of personnel,
- Entrance of a person not qualified for a present hazard,
- Occurrence of a hazard requiring the absence of personnel in presence of personnel,
- Occurrence of a new hazard in presence of unqualified personnel.

Two different kinds of safety systems are required to prevent these transitions:

- Access control to the building and to its different areas, that involves doors switches, safety locks and associated hardware.
- Risk management that involves safety interlocks, in relation with the potentially hazardous equipments. These equipments have to wait for permissive before generating any hazard, and have to acknowledge when the hazard is present.

FUNCTIONNAL ARCHITECTURE DESIGN

Design Principles

To satisfy at the lowest cost the requirements of safety regulations and those of the operation management, the choice was made to implement a functional architecture built around two independent technological barriers when required by the risk level.

The combination of these two independent technological barriers allows managing the dynamic evolution of the compromise between hazard presence and worker presence in the rooms, throughout the various scenarios identified in the safety studies.

Each technical barrier is composed of two subsets, one dedicated to hazard sources management, and the other one dedicated to worker presence management.

The two completely independent barriers, even at the sensor or actuator level, are designed with different technologies adapted to the required Safety Integrity Level (SIL 2 or SIL 3). The combination of these 2 barriers is equivalent to a unique barrier with a rate of dangerous failure of ~10^{-6} per year.

IEC 61508 Standard

The **IEC** (International Electrotechnical Commission) 61508 standard specifies a set of requirements for functional safety of electrical / electronic / programmable electronic safety-related systems.

It defines 4 levels of requirements or « SIL » that must be respected according to the acceptable failure objective of a safety function, either in continuous operation mode, or in low demand operation.

These levels are used to specify the safety requirement of each device or software involve in the SSP.

Table 3: SIL v.s. Average Failure Probability

Safety Integrity Level	Average probability of Failure on Demand per year
SIL 4	$\geq 10^{-5}$ à $< 10^{-4}$
SIL 3	$\geq 10^{-4}$ à $< 10^{-3}$
SIL 2	$\geq 10^{-3}$ à $< 10^{-2}$
SIL 1	$\geq 10^{-2}$ à $< 10^{-1}$

FUNCTIONAL DESCRIPTION OF SUBSETS

The SSP is composed of 3 main subsystems:

- The first technical barrier (TB #1),
- The second technical barrier (TB #2) totally independent from the first one,
- The SSP supervisory system that present different GUIs to the operator.

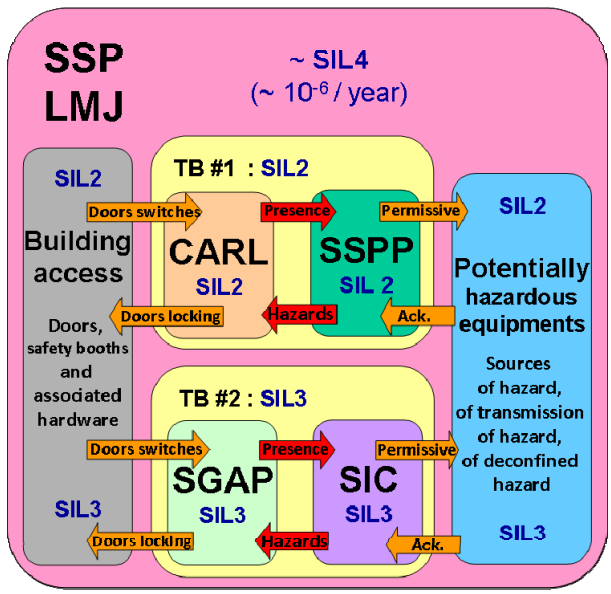


Figure 1: General SSP architecture.

First Technical Barrier

The first barrier designed in SIL 2 is based on a programmable technology (safety PLC). It is itself composed of two functional subsets:

- The "CALR" (Contrôle d'Accès des Locaux à Risques) ensuring access control to the areas that present hazards, by using contactless personal badges (RFID technology) and safety locks.
- The "SSPP" (Système pour la Sécurité du Personnel Programmé) controlling both the presence of hazards and their authorizations (permissive) at the equipment level.

The probability of a dangerous failure of the first barrier is between 10^{-2} and 10^{-3} per year.

Second Technical Barrier

The second barrier designed in SIL 3 is based on a non programmable technology (safety relays or equivalent). It is itself also composed of two functional subsets:

- The "SGAP" (Système de Garantie d'Absence de Personnel) whose objective is to ensure the absence of workers in the target bay during a powerful laser shot, thus preventing the risk of death due to a neutron flash. Access management to the rooms is done using access keys provided by a guard.

- The "SIC" (Système d'Interverrouillage Centralisé) that ensures, with key based safety interlocks, that the laser beams and the power conditioning system cannot be activated unexpectedly during a maintenance period.

The probability of a dangerous failure of this barrier is between 10^{-3} and 10^{-4} per year.

SSP Supervisory Software

Dedicated GUIs are provided to the operators in charge of the LMJ safety.

The main GUI presents the status of all the LMJ SSP. It has an alarm zone and an event log to allow alarm managing, and different control zones with push buttons to deliver risks authorization. Other GUIs present to the operator a general view of risks, a general view of the process state and a general view of the building security. Detail views are dedicated to hazardous equipments such as laser bundles, power conditioning devices, and Laser sources.

This software layer is designed with Panorama E^2 (the CODRA Company SCADA product) under Windows 7. It is independent from the safety loops which are controlled at the lowest level by PLC.

REFERENCES

[1] IEC 61508 international standard (International Electrotechnical Commission, 3, rue de Varembé Geneva, Switzerland)

[2] The Laser Megajoule facility: control system status report, ICALEPCS 2007, by J.P. Arnoul, F. Signol CEA/CESTA, Le Barp, 33114 France, P. Bétrémieux, J.J. Dupas, J. Nicoloso CEA/DIF, Bruyères le Châtel, 91680 France

[3] The Laser Megajoule facility: control system status report, ICALEPCS 2009, by J.J. Dupas, J. Nicoloso CEA/DIF, Bruyères le Châtel, 91680 France

AUTOMATIC ANALYSIS AT THE COMMISSIONING OF THE LHC SUPERCONDUCTING ELECTRICAL CIRCUITS

H. Reymond, O. O. Andreassen, C. Charrondiere, A. Rijllart, M. Zerlauth,
CERN, Geneva, Switzerland

Abstract

Since the beginning of 2010 the LHC has been operating in a routinely manner, starting with a commissioning phase and then an operation for physics phase. The commissioning of the superconducting electrical circuits requires rigorous test procedures before entering into operation. To maximize the beam operation time of the LHC, these tests should be done as fast as procedures allow. A full commissioning need 12000 tests and is required after circuits have been warmed above liquid nitrogen temperature. Below this temperature, after an end of year break of two months, commissioning needs about 6000 tests. As the manual analysis of the tests takes a major part of the commissioning time, we automated existing analysis tools. We present here how these LabVIEW™ applications were automated, the evaluation of the gain in commissioning time and reduction of experts on night shift observed during the LHC hardware commissioning campaign of 2011 compared to 2010. We end with an outlook at what can be further optimized.

INTRODUCTION

The first LHC Hardware Commissioning Campaign (HCC) was started early 2008. The aim of this crucial phase, mandatory before to put the accelerator into operation, was to test in real conditions all the equipment and systems of the machine. It included protection systems and all the superconducting electrical circuits.

For this purpose, two types of tools were developed [1], one to manage the execution of the tests and a second to provide data analysis, to deal with the big amount data from the tests. The initial approach used when designing these analysis tools, was to help the users to perform manual analysis of the data with a tool that permitted to select a test result set, automatically load interesting signals and display them in a GUI that could easily be used by experts from several domains.

During the HCC, human and time resources have been optimized. Less people were available for the LHC start-up tests therefore to speed up the tests, the framework and the applications have been improved to permit execution of tests in parallel on several circuits. Furthermore, with the increased electrical circuit's knowledge, it has been possible to set limits to measured and calculated parameters, which led to automate part of the analysis.

SYSTEM OVERVIEW

The Electrical Circuit Commissioning (ECC) consisted of a series of tests to provoke a reaction of all interlock and protection systems. Different current cycles (see Fig.1) were applied on these circuits to validate them, up to nominal current.

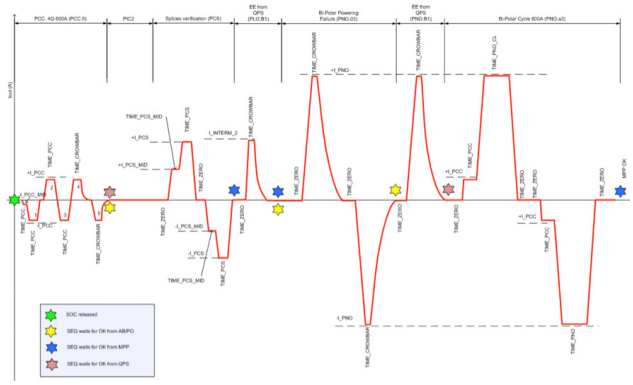


Figure 1: Current cycles for 600A circuit validation.

Each reaction of a protection system generated a so-called "Post Mortem" file, in which the most important parameters were stored in a circular buffer with time stamped data. These files were then collected and archived to be later analysed using the Post Mortem Analysis tools.

The Post Mortem Analysis System

The Post Mortem framework is made of three main systems [2]. The *Sequencer* application edits test cycles and sends commands to the power converters. The *Post Mortem Request Handler* (PMRH) collects the data files from the involved equipment and associates them to the test parameters sent by the *Sequencer*. The *Post Mortem Event Analyser* (PMEA) provides a GUI for experts to easily retrieve and analyse a test.

Several GUIs have been built to present data of specific tests. They display the executed current cycle, the values returned by the involved equipment and the calculated values (ramp, min-max, delays, resistances...).

Principle of Validation of an Electrical Circuit

The validation of an electrical circuit is performed as follows: From the sequencer, an operator selects the circuit to be tested and attributes a series of test steps to be performed, depending on the circuit type (nominal current and specific limits). Automated checks are made to verify that the circuit is available and not blocked by interlocks. When all conditions are met, the sequencer generates and sends commands to the power converter of the related circuit.

During the current cycle execution, the protection equipment reacts and generates post mortem files. These files are collected by the PMRH, from all systems in the

interlock and protection chain of the circuit. Their data are merged into one event. Then all available events are presented in the PMEA GUI. It gives an overview of the executed tests that are waiting for analysis, but also those already verified.

To validate a test step of a circuit, the event generated must be analysed and signed by experts of all involved domains (interlock system, powering, quench protection, cryo). When all the experts have signed and accepted the analysis results, this information is returned to the *Sequencer*. This allows the system to launch the next test step. In case of rejection of one analysis, the circuit powering is blocked.

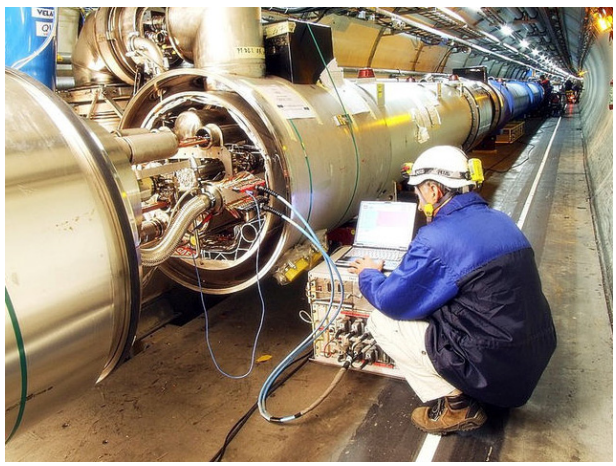


Figure 2: High Voltage qualification in the LHC tunnel.

Then the problem must be investigated and solved before a new test sequence is launched. This could means an intervention in the LHC tunnel (see Fig. 2) to make on site measurements to understand the facts, replacement of some parts in case of device failure or sometimes loading a new firmware release into an electronic system.

Each tunnel intervention must be scheduled and well organized to optimize the time and the work of the involved teams.

LHC Circuits Layout

The LHC circuits are divided into eight independent sectors. Each circuit can supply one or many magnets, depending of the circuit type. For example, one 13kA power converter drives 154 dipole magnets. There is a large range of different circuits, with currents of 60A, 80A, 120A, 600A, 6kA, 8kA and 13kA. These circuits power all the main dipoles and quadrupoles and their respectives correctors, such as sextupoles, octupoles, decapoles and dodecapoles.

ELECTRICAL CIRCUIT COMMISSIONING EVOLUTION

Almost all of the test management and analysis tools used today have been developed for the first complete Hardware Commissioning of the machine in 2008. Many improvements were made during the next campaigns, thanks to the experience gained. But also an effort to enhance the efficiency was conducted in order to minimize the testing phases, with the objective to increase the time dedicated to physics.

In the early age of the ECC, the sequencer could only manage one test at a time. It was then modified [3] so that it can run tests on various circuits in parallel. Introducing this feature in the test sequences was possible due to the sectorisation, but also by the large range of circuit types.

To meet this challenge we instantiated some parts of the PMRH code, used for collecting the PM files. As the PMRH, the PMEA and all analysis tools were developed with LabVIEW™; it was easy to introduce VI templates, and then call them as a clone when needed. From the analysis framework point of view, introducing parallelism in the tests was transparent.

Taking into account the Human Factor

At that stage the experts, started to be drowned in hundreds of events per day; to analyse and sign. In effect, without counting the selection tools for viewing the tests and those dedicated to the display of data as curves with pre-loaded signals, it took about 3 minutes to analyze a test. With an average of 300 tests per day to achieve the objectives, 15h in front of the screen were necessary, to check similar plot and table results. As the slightest variation could have an enormous importance in terms of reliability and safety of the machine.

As for any major achievements, taking or not into consideration the human factors that may be crucial for the success of the project. In the case of ECC, for many technical fields, the same experts who participated in the design and installation of the system then participated in validation tests, and sometimes also applying hardware corrections and compliance. It is not difficult to understand that whatever the motivation and professional commitment of the person who analyzes, his acuity and assessment will not be the same between the first and the 200th analysis. So, even by applying the same criterias, the first bad events will be surely rejected, but later could be accepted as near to the limits.

Introducing Automated Analysis

To overcome the problem of the number of tests to be signed by the experts, it was decided to automate some analysis. This was made possible from the ECC 2009, using the knowledge acquired from the previous campaign. Hence some acceptance ranges and even more restrictive criterias could be defined to specify automatic analysis algorithms for some particular circuits.

The first tool fitted with automated analysis was the Powering Interlock Controller (PIC) [4]. Originally it was dedicated to verify delays and status of digital signals, coming from the interlock system. The role of the expert beiing to check some patterns and signals synchronisation, according to various parameters, such as circuit type, test type and protection equipment present in the interlock loop.

After one complete ECC, expected delays and synchronisation scenarios were well known. They have then been coded in LabVIEW™ and introduced in a new PIC analysis (see Fig. 3). The left part shows interesting digitals and analog signals; the right side is dedicated to a summary table, with measured and expected values. Overranged values are highlighted in red. The bottom area is reserved for the final analysis result represented by a big passed / failled indicator and a user comments field. A button allow experts to sign the test (accept or reject) after authentication.

The PIC analysis tool was a good candidate to start automatisation. Once digital signals patterns and delays are measured and valided by the interlock equipment experts, they remain almost fixed for the system lifetime. Second, the PIC systems commissioning needs repetitions of the same test on large number of circuits. So automatising is a real benefit.

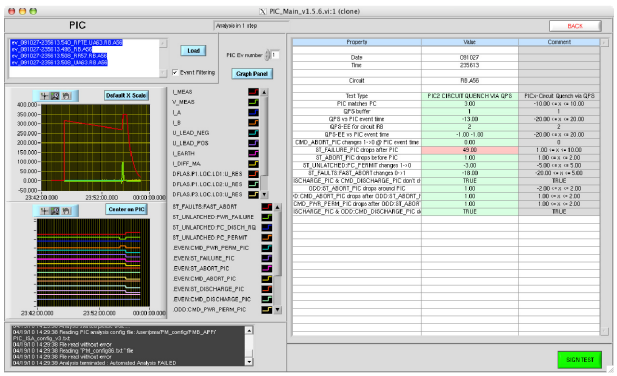


Figure 3: Analysis panel for PIC2 test type.

Following this example, two other analysis have been automated. The Powering to Nominal used to measure and check a set of values during some current cycles execution (resistance, ramp rate, overshoot). The Discharge analysis aims to study power converter reactions that receive a slow or fast power abort signal. For these two applications, useful signals were presented as XY graphs and compared to expected criterias, displayed in a concise results table.

Full Automated Analysis and Test Signing

A last step in improving the efficiency of ECC has been developed and used at the beginning of the LHC HWC 2011, with the automated execution and signing of the analysis.

The analysis tools, with automated verification procedures, allows on one hand to verify the proper functioning of these tools (application of criteria adapted to the test conditions) and on the other hand to use the confidence in the system acquired during the validation campaigns of electrical circuits in 2009 and 2010.

From a technical point of view, the PMA applications have been modified to incorporate the automation. A selection tool has been added to the PMEA, to select which test types will be automatically analyzed and signed, in the three applications involved (PIC, PNO and Discharge).

On the PMRH side, the buffer collection was configured thanks to a look-up table, that associates each test type with the expected number of buffers for each equipment covered by the test. Hence, once all the data are received, the required analysis for an event can start immediatly. The PMRH being a server, it is possible to run in parallel dozens of analysis.

For the three analysis functions involved, we added them to a new execution mode. So when a call for an "automated-signing" is defined by the PMRH, the analysis runs, the results are generated and the signature (passed or failed) applied by the program. The final result of the test and the parameters of the event are then put at disposal to the experts, via the GUI of the PMEA.

They are also forwarded to the *Sequencer*, which then could block the electrical circuit in case of rejection, or launch the next stage of the test, if accepted. All these steps are executed transparently for the users, only failed analysis are highlighted in the PMEA, waiting for a deeper investigation by an expert.

Validating the Validation Tools

One of the crucial aspects of an automatic analysis is its integrity and reproducibility. Once the expert is confident with the tool, it is clear that he/she will not spend time looking at the passed analysis. Operation and safety of the machine depends then on the software quality.

To check these criterias, reference tests were selected by different experts for each type of analysis. Thus it is possible to restart as many times as necessary the tests (with known results) to verify the compliance of the results. A special module has been developed in the PMRH, it allowed us to quickly validate our tools at the beginning of an ECC, after modification, or in the case of a change of the LabVIEW™ version.

PRESENTATION OF THE RESULTS

The first two Hardware Commissioning Campaigns were a complete machine commissioning. In 2008, before the first exploitation of the machine, 15532 tests have been executed during 164 days, among them 12666 were successful. In 2009, after the incident that occured on September 19th 2008, 13611 tests have been performed in 89 days, with 11175 successfull. This moved the efficiency up to 82.10%.

For the following campaigns, the allowed timescale was drastically reduced to increase the time dedicated to physics operation. Moreover, most of the electrical circuits were not warmed above 80K during the last technical stops, it was then possible to reduce the number of tests, according to the circuits history.

In 2011 the ECC has been done in 21 days, with 6092 tests executed. From these, 3204 were automatically signed, 2065 by PIC analysis and 1139 by PNO. But still 1101 test for PNO.d1 and 496 for PNO.d3 had to be manualy signed with the Discharge analysis tool.

Several Sources of Improvement

Since the 2010 ECC, new valuable features have been integrated in the PMA framework: automated analysis, accepted tests results directly sent to the MTF (Manufacturing and Test Folder repository of the related circuit). This gives a full traceability of circuit behaviour, without possible error due to human input. On the other side, for failed tests, a non-conformity report is generated and forwarded to the dedicated management tool. This gives to the expert in charge an overview of the encountered defects, but also a trace of what has to be fixed, reducing the risk of missing one issue.

The time saved thanks to the signature test automation, freed the experts for critical tasks such as monitoring the repair of problems. This also allowed reducing the number of persons involved in the test execution.

The change in the organization of the tests [5] has been an important contribution too. The availability of night shifts launching a large amount of sequences of tests during the night, on all sectors in parallel gives the field workers the opportunity to organize and fix problems on non-compliant systems the following day.

CONCLUSION AND PERSPECTIVES

The Electrical Circuit Commissioning of the LHC is now a well-defined and documented exercise fitted with powerful procedures and software.

The automated analysis and results signing have proven their utility and efficiency. Reducing the time devoted to this task, helped to reorganize the experts' responsibilities and to decrease the number of people implicated.

A big effort has been done to take into account the human factor, when developing the analysis framework. Some improvements have still to be done, for user training, as specification of the tools are not directly written by future users, lots of practical features and useful tips included in the tools are unknown to them.

Most of the electrical circuits have been qualified to 3.5 TeV operation. During the next 2013 long shutdown, their qualification to 7 TeV (i.e. nominal magnet current) will require extended sequences of tests. Here again, fully automated analysis tools will be a key point for an efficient ECC execution. The future LHC operation needs more powerful applications to study complex conditions and machine interaction. They will help operators to overcome difficult behaviour and to get a forecast about equipment and systems status.

REFERENCES

[1] M. Zerlauth et all, "The LHC Post Mortem Analysis Framework", ICALEPCS 2009, Kobe, Japan.

[2] O. O. Andreassen, "Post Mortem for the LHC", EN/ICE Workshop on Post Mortem, 2009, CERN, Switzerland.

[3] B. Bellesia et all, " Optimisation of the Powering Tests of the LHC Superconducting Circuits", PAC09, Vancouver, Canada.

[4] M. Zerlauth, R. Schmidt, J. Wenninger, "Commissioning and Operation of the LHC Machine Protection System", HB2010, Morschach, Switzerland.

[5] M. Solfaroli, "Scope and Results of Hardware Commissioning to 3.5 TeV and Lessons Learnt", Chamonix 2010 Workshop, Chamonix, France.

AUTOMATIC INJECTION QUALITY CHECKS FOR THE LHC

L. N. Drosdal, B. Goddard, D. Jacquet, R. Gorbonosov, S. Jackson, V. Kain, D. Khasbulatov, M. Misiowiec, J. Wenninger, C. Zamantzas, CERN, Switzerland.

Abstract

Twelve injections per beam are required to fill the LHC with the nominal filling scheme. The injected beam needs to fulfill a number of requirements to provide useful physics for the experiments when they take data at collisions later on in the LHC cycle. These requirements are checked by a dedicated software system, called the LHC injection quality check. At each injection, this system receives data about beam characteristics from key equipment in the LHC and analyzes it online to determine the quality of the injected beam after each injection. If the quality is insufficient, the automatic injection process is stopped, and the operator has to take corrective measures. This paper will describe the software architecture of the LHC injection quality check and the interplay with other systems. Results obtained during the LHC run 2011 will finally be presented.

INTRODUCTION

The LHC is filled through two transfer lines (TI2 and TI8) from the last pre-injector, the SPS. Currently the LHC runs with 1380 bunches per beam, which require 12 injections (plus one low intensity bunch) per ring. The maximum number of bunches per injection is 144 corresponding to 1 MJ stored energy.

Injection is a complicated process. The correct number of PS batches has to be injected in the correct RF bucket in the LHC. The injector timing system [1] and the LHC RF system ensure LHC dynamic injection requests and synchronisation. The extraction event from the SPS is forwarded to the LHC timing system as LHC injection event to trigger beam instrumentation.

The series of required injections is pre-programmed as an injection sequence driven by the injection sequencer. According to the filling scheme requests are sent to the injector timing system. At each injection the injection quality check (IQC) analyses data from key equipment in the LHC and the transfer lines and provides a result instructing the injection sequencer how to proceed: continue, stop or repeat the same request.

The IQC also provides software interlocks which are picked up by the software interlock system (SIS) and inhibit injections. A GUI for visual display of the detailed result is provided for monitoring and operator interaction, see Fig 1. The interplay of the IQC with other systems is shown in Fig. 2. There is also a playback tool for reviewing events. This paper presents the IQC architecture and results from the LHC run 2011.

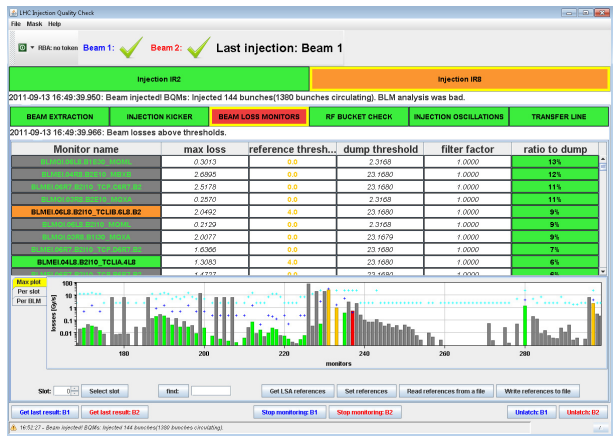


Figure 1: IQC GUI for monitoring and operator interaction. BLM panel showing loss distribution over the injection region.

LHC INJECTION QUALITY CHECKS

The injection quality check analysis is triggered automatically at each LHC injection event. Only data which has a time stamp within a certain time window around the injection time stamp are accepted. This window is set individually for each system and ranges from 1 s before the injection to 6 s after.

The analysis is composed of separate analysis modules which are launched as soon as the required data is ready. The different modules are combined to an overall result at the final stage.

Data collection takes on average 4.5 s and analysis about 0.5 s. In case of missing data the analysis automatically times out after 10 s. The injection sequencer and IQC analysis are independent for beam 1 and beam 2. In this way the next injection can already be requested while the analysis on the other beam is still ongoing.

IQC Architecture

The IQC uses the LHC post mortem framework for data storage and analysis [2]. Data collection as well as analysis are written in JAVA and are running on JAVA servers. The results are published through Java Messaging System (JMS) and are picked up by the by the injection sequencer, the SIS and the IQC GUI.

For threshold management the IQC analysis uses the LSA Management of critical settings [3]. The thresholds can be modified by experts in the IQC GUI.

Event data stored on the post mortem server can be replayed offline on a separate, but identical application. This feature can be used for diagnostics of previous events, for testing and to make statistics.

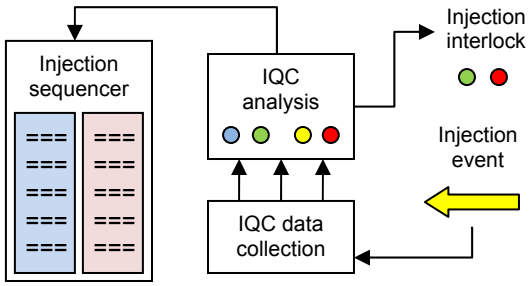


Figure 2: Figure showing the interplay of the IQC with injection sequencer, the SIS (Injection Interlock) and timing system. The analysis is triggered by the injection event. The overall result is published to the injection sequencer, and flags are produced for the SIS.

ANALYSIS MODULES

Correctly Filled Bunch Pattern

Not every injection request is successful. Due to bad beam quality beam might not even be extracted from the SPS. To determine whether the beam was injected into the LHC, the results of the upstream and downstream BCTs in the transfer lines are combined with the BQM result. Three devices are used for the analysis due to reliability issues of the BCTs in the transfer lines. If the devices disagree the overall outcome of the IQC is UNKNOWN.

The LHC BQM uses the wall current monitors to find the longitudinal positions of the injected bunches [4]. In the module RF bucket check, the IQC compares this information with the requested filling pattern. In case of inconsistency the IQC stops the injection sequencer.

Injection Kicker Checks

The LHC injection kicker system (MKI), consisting of four vertical kicker magnets per beam, needs to have a short kicker rise time of less than 1 μs, little ripple on about 8 μs long flattop and fall times not longer than 3 μs, [5]. The characteristics of the injection kicker waveforms generated by oscilloscopes in the tunnel are analysed by the IQC after each injection. With the tight IQC thresholds deterioration of the kicker characteristics are noticed immediately.

Beam Losses

The module which is used mostly and provides very useful information for injection tuning is the beam loss module. At each injection the beam loss monitor crates of interest are triggered to produce a special buffer for the IQC, consisting of 512 beam loss samples per monitor around the injection event with 40 μs integration times.

The transfer line collimators (TCDIs) are at the end of the lines and losses on these are seen on the BLMs of the LHC superconducting magnets. The BLMs have low dump thresholds and small losses on the transfer line collimators can already lead to a beam dump while filling the LHC. The loss profile from the transfer line collimators is a valuable diagnostic for beam quality problems in the transverse plane.

Also the losses around the protection devices (TDI, TCLIa, TCLIb) protecting against injection kicker failures are shown. Losses from uncaptured circulating beam, satellites or uncaptured beam from the injectors as well as nominal beam in case of kicker problems ends up on these. The loss signature for the different cases is typical and is used for problem identification. Fig. 3 shows the loss distribution plot from the IQC BLM panel.

In addition to the LHC ring data the beam loss monitors in the transfer lines are also recorded.

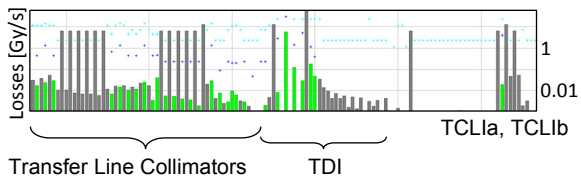


Figure 3: Distribution of beam losses as seen in the IQC GUI. The beam loss pattern is used to indicate the source of losses.

Injection Oscillations and Transfer Line Trajectory

The setting of all the protection devices in the LHC is valid for a given aperture in the machine. The settings have to include tolerances for orbit distortions, energy errors, beta beat and injection oscillations. If the injection oscillations are larger than the tolerance, protection against beam impacting the aperture during e.g. injection kicker errors cannot be guaranteed.

Orbit information as well as turn-by-turn data from BPMs triggered at injection is acquired by the IQC in the LHC injection regions. The injection oscillation amplitudes have to be below the IQC limit of 1.5 mm (1.75 mm for TI 2 H).

If the limit is exceeded, the SIS inhibits injection of high intensity for that particular beam. A special flag is provided by the IQC for the SIS for that purpose. Low intensity (about 10^{12} protons) can be injected to correct. The flag is automatically reset as soon as the injection oscillations are within limit again.

The reading of the BPMs in the transfer lines are also recorded in the IQC. The trajectory offsets in the transfer line collimators have to be minimised to reduce losses.

2011 EXPERIENCE

Since the first running period in 2010 the IQC analysis has become an integral tool of understanding injection problems and optimising injection efficiency. The beam loss monitor results are used routinely.

Statistics From Mid July to Mid August 2011

60 LHC fills from mid July to mid August 2011 were analysed. Within this period 1483 IQC analyses were triggered. The IQC latched 7.9% of the time indicating quality issues. For 11.7% of the events the expert warning result was given due to beam losses. The full distribution

of results is given in Fig. 4. For a detailed distribution of failures see Fig. 5.

During the period of these 60 fills the injections were very clean. The beam scraping in the SPS was used at maximum to cut beam tails. No other injection issues occurred. The less than 1 % of IQC result "UNKOWN", associated with missing data, indicate the good performance of the overall system.

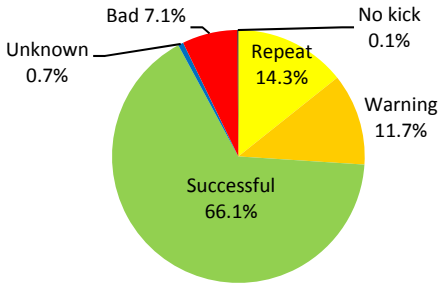


Figure 4: Distribution of IQC results over a period of 60 fills. In total there were 1483 injection events, where 7.9% latched.

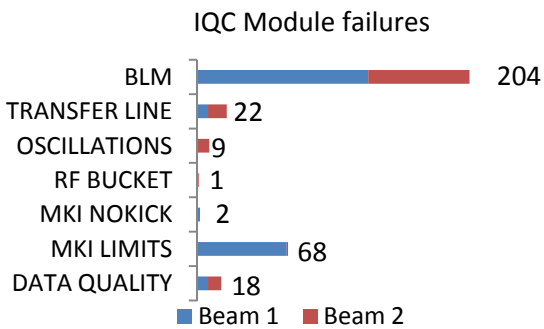


Figure 5: Distribution of module failures over 1483 injections; beam losses, transfer line (BLMs and BPsM), injection oscillations, RF bucket check, MKI not pulsed, MKI waveform out of thresholds, bad data quality. The total number is given right of the plot, the bars indicate the distribution between beam 1 and beam 2.

Data Collection Issues

The IQC is depending on a large number of data sets from many sources. In case of missing data the IQC automatically stops injection. During the commissioning run of 2010 events with missing data were frequent. All data collection problems could be solved during the winter shutdown and the beginning of the LHC run in 2011. The remaining issue is the data quality from the transfer line BCTs we still frequently have issues with the data quality. During the analysed period of above for 47.9% only one of the BCTs produces data or the two BCTs give inconsistent data. For the total of 60 fills analysed for only 18 events (1.2%) a data set was missing and caused an injection interlock.

Beam Quality Issues Discovered

In this section examples are presented where beam quality issues could be discovered due to the IQC results.

At the beginning of injecting full SPS batches into the LHC, consisting of 144 bunches, spaced by 50 ns, the IQC indicated large injection oscillations in the vertical plane for beam 1, see Fig. 6. It turned out that the injection kick length had not been long enough and some of the beam was kicked on the falling edge of the waveform. It was subsequently prolonged.

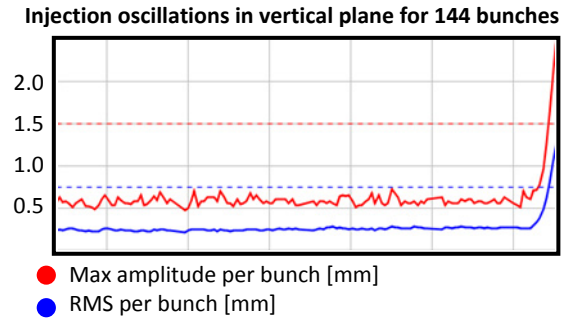


Figure 6: Injection oscillations in the vertical plane for 144 bunches. The last bunches had larger oscillation amplitudes due to too short kicker waveform. [23:21:38 20/6-2011]

The transfer lines suffer from trajectory stability problems in the horizontal plane [6]. Slow drifts, shot-by-shot and also large bunch-by-bunch variations have been recorded. The IQC injection oscillation analysis was very useful to detect the large bunch-by-bunch differences for beam 2 H, see Fig. 7. The source for these is probably the horizontal extraction kicker in the SPS. The issue is still under investigation.

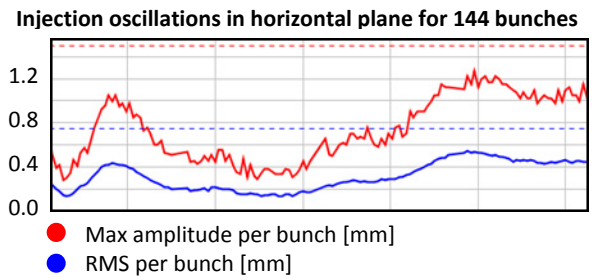


Figure 7: Injection oscillations amplitudes in the horizontal plane for 144 bunches for beam 2. A bunch-by-bunch variation of more than 1 mm was discovered. [01:28:57 25/9-2011]

On 18th of April 2011, 11 magnets were quenched during a high intensity injection attempt when an injection kicker flashover of one of the four beam 2 magnets occurred. The problem could be diagnosed immediately with the IQC MKI analysis indicating the shortened waveform for one of the four kicker magnets, see Fig. 8.

The IQC BLM buffer is now also used for studying very fast beam loss phenomena occurring shortly after injection, called UFOs [7].

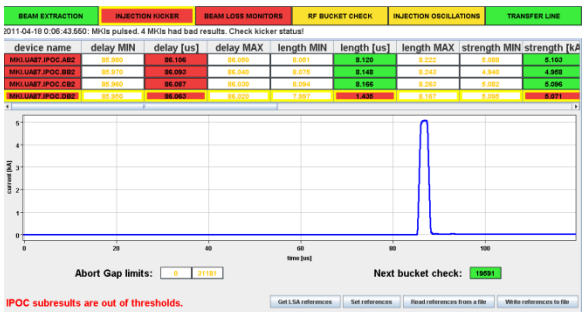


Figure 8: IQC MKI panel showing the MKI waveform of magnet D which had a flashover and had a length of about 2μs only instead of the required 8 μs.

CONCLUSION

The LHC injection quality check has become an important part of the LHC software suite ensuring good quality beams and injection protection. The chosen modular architecture based on the post-mortem framework has proved valuable and allowed stepwise commissioning and easy diversification of the analysis.

The IQC analysis is used offline and online on a routine basis to tune and improve injection. It has played a major role to achieve routine LHC filling with 1 MJ beams.

REFERENCES

[1] J. Lewis et al., "The Evolution of the CERN SPS Timing System for the LHC Era", ICALEPS'03, Gyeongju, Korea, 2003.

[2] M.Zerlauth et al., "The LHC Post Mortem Analysis Framework", ICALEPS'09, Kobe, Japan, 2009.

[3] W. Sliwinski, P. Charrue, V. Kain, G. Kruk, "Management of Critical Machine Settings for Accelerators at CERN", ICALEPS'09, Kobe, Japan, 2009.

[4] G. Papotti et al, "Stability Longitudinal Beam Measurements at the LHC: The LHC Beam Quality Monitor ". IPAC'11, San Sebastion, Spain, 2011.

[5] O. Bruning et al. (Eds), "LHC Design Report, Vol. I, The LHC Main Ring", Chapter 16, CERN-2004-003, CERN, 2004.

[6] V. Kain et al, "Stability of the LHC transfer lines". IPAC'11, San Sebastion, Spain, 2011.

[7] T. Baer, "UFOs in the LHC", IPAC'11, San Sebastian, Spain, 2011.

FIRST EXPERIENCES OF BEAM PRESENCE DETECTION BASED ON DEDICATED BEAM POSITION MONITORS

A. Jalal, S. Gabourin, M. Gasior, B. Todd, CERN, Geneva, Switzerland

Abstract

High intensity particle beam injection into the LHC is only permitted when a low intensity pilot beam is already circulating in the LHC. This requirement addresses some of the risks associated with high intensity injection, and is enforced by a so-called Beam Presence Flag (BPF) system which is part of the interlock chain between the LHC and its injector complex. For the 2010 LHC run, the detection of the presence of this pilot beam was implemented using the LHC Fast Beam Current Transformer (FBCT) system. However, the primary function of the FBCTs, that is reliable measurement of beam currents, did not allow the BPF system to satisfy all quality requirements of the LHC Machine Protection System (MPS).

Safety requirements associated with high intensity injections triggered the development of a dedicated system, based on Beam Position Monitors (BPM). This system was meant to work first in parallel with the FBCT BPF system and eventually replace it. At the end of 2010 and in 2011, this new BPF implementation based on BPMs was designed, built, tested and deployed.

This paper reviews both the FBCT and BPM implementation of the BPF system, outlining the changes during the transition period. The paper briefly describes the testing methods, focuses on the results obtained from the tests performed during the end of 2010 LHC run and shows the changes made for the BPM BPF system deployment in LHC in 2011.

INTRODUCTION

A high intensity particle beam injection into the LHC must not be permitted if no low intensity beam is already circulating, confirming the set-up of the primary machine parameters, like beam orbit, operational point, chromaticity, etc. For this reason, the extraction and injection process of beam from the SPS to the LHC is tightly interlocked. Two BPF flags, one for each beam, are generated and transmitted to the extraction interlock systems indicating the presence, or absence, of a circulating beam in the LHC.

In the 2008, 2009 and 2010 runs each flag was derived from the system measuring the beam currents passing through the FBCTs installed on the LHC beam pipes. The FBCT system was designed for beam instrumentation purposes and could not fulfil all rigorous requirements of the LHC MPS. To increase the reliability of this important protection feature a dedicated BPM system has been developed in addition to the FBCT solution. Finally, in the spring of 2011 the FBCT BPF system was phased out in favour of the dedicated BPM implementation.

SYSTEM ELEMENTS

The BPF system uses signals from the FBCT and BPM systems. The BPM BPF system [1] is based on four signals per LHC beam derived from electrodes of a dedicated BPM. Each electrode drives an input of a dedicated BPF analogue front-end [1], which in turn derives Boolean beam presence flags. For beam-1 these flags are BPF_1A, 1B, 1C and 1D, for beam-2: BPF_2A, 2B, 2C and 2D. The BPF front-end sets a flag to TRUE if the corresponding signal exceeds a threshold and the beam has been already circulating for certain time.

A pair of FBCTs also evaluates beam presence based on the circulating beam characteristics, deriving BPF_1E and 1F for beam-1, 2E and 2F for beam-2. This gives a total of six flags per beam as illustrated in Fig. 1.

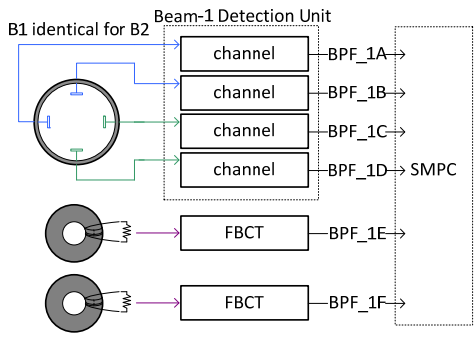


Figure 1: System Elements.

These flags are then transmitted to the Safe Machine Parameters Controller (SMPC). The SMPC logic carries out a voting on the six input flags using a pair of two-out-of-three (2 oo 3) majority voter blocks and also filters the signals to remove spurious transitions. This results in a pair of Beam Presence Flags per beam which are transmitted to the extraction Beam Interlock System (BIS). This pair is also combined to give a single BPF which is broadcast for general consumption via the LHC General Machine Timing (GMT), as sketched in Fig. 2.

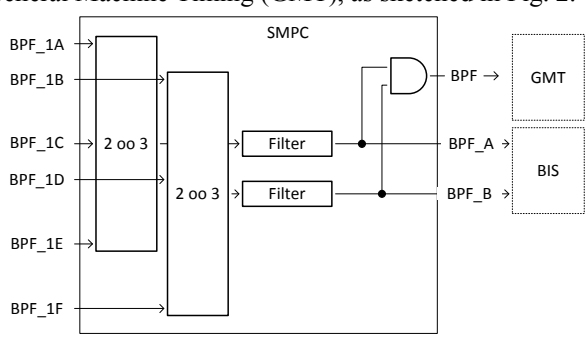


Figure 2: Combination and Voting Logic.

TESTING AND CHARACTERISATION

The BPM implementation was tested in parallel with LHC operations in late 2010. This allowed the system to be characterised with respect to the FBCT.

The test consisted of comparing two of the BPF from the BPM (BPF_1A and BPF_1B) with intensity values during ordinary LHC operations with beam. The intensity values were given by the LHC FBCT, the BPF values were logged by the BIS. The test period covered five weeks of 2010 operation with beam (25/10 – 19/11).

As the LHC was operating normally with beam, there were a large amount of transitions. For the analysis, the key area of interest was the intensity value at the moment of a BPF transition from TRUE to FALSE, or from FALSE to TRUE. Figure 3 shows typical characteristics of intensity, threshold and BPF transitions.

Threshold During Normal Operation

The beam intensity was recorded at the time of every flag transition; this resulted in a consistent value of intensity with an average around 3 x 109, as shown in Table 1. There are, however, two important limitations in this test method: The Pick-up selection, and the FBCT accuracy.

The pick-ups being used for the characterisation were those on the upper and left side of the beam pipe as shown in blue in Figure 1. The signal amplitude on each BPM electrode depends on the distance of the beam to the electrode, so the signals are only equal for a centred beam. The minimum intensity threshold values observed are related to a beam that was circulating closer to the BPM electrode, resulting in a larger signal. On the other hand, the maximum threshold intensity values are due to a beam circulating further from the BPM electrode, thus giving a smaller signal. These results are summarised in Table 1.

Table 1: Thresholds for the BPM BPF system

	Beam 1	Beam 2
Min value	2.11×10^9	1.75×10^9
Max value	6.05×10^9	6.33×10^9
Average	3.3×10^9	3.5×10^9

On average, the beam intensity at the transition point, with significant position dependence, is around 3×10^9 charges. The observed intensity threshold dependency on beam position was removed for the 2011 implementation [1].

The second limitation with this method is that it followed normal LHC operation, where the beam is injected or dumped in a single turn, with the threshold value given from the FBCT reading at that instant. The threshold value determined in this manner is dependent on the measurement algorithm and time delays of the FBCT system. Therefore a dedicated run was organised, to determine the threshold of the BPM system with a higher precision.

Threshold During Dedicated Run

Slowly decreasing the beam intensity proved the best method for determining the threshold, as the threshold is passed, oscillations of the BPFs occur. Oscillations like this are the best indicators that the beam is very close to the threshold value of the dedicated BPMs. As shown in Table 2, the threshold was determined to be around 6×10^8 charges giving rise to a noise around 2 kHz.

Table 2: Low Intensity Noise

Flag	Time	Intensity	Frequency
Beam 1	04:15:43.956	5.9×10^8	2.1 kHz
Beam 2	20:19:12.617	5.9×10^8	2.4 kHz

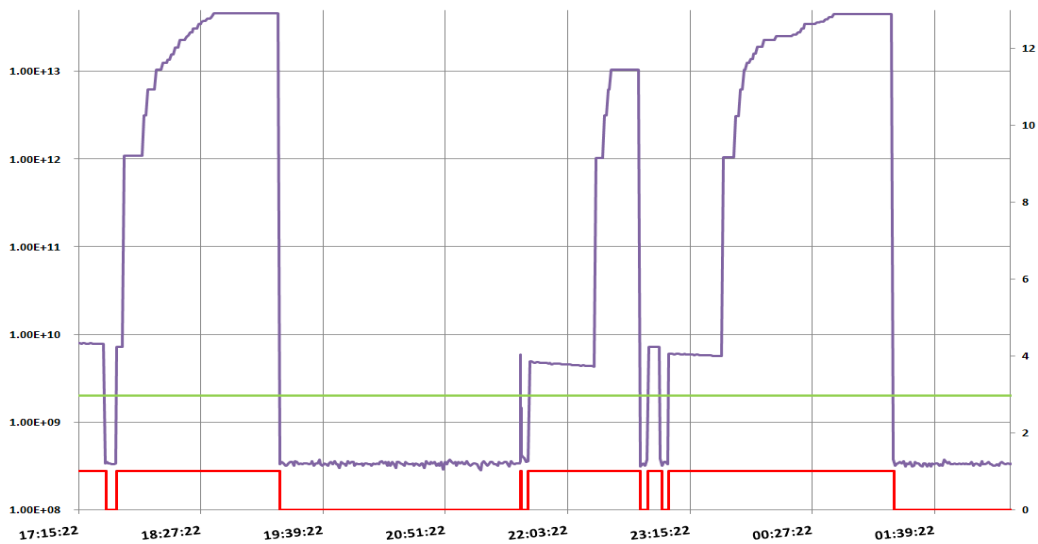


Figure 3: Intensity (Purple), Threshold (Green) and Beam Presence Flag (Red) Normal Operation.

This noisy behaviour was only observed in the case of *decreasing* intensity, and not in other situations. This is due to the noise being only observable when the intensity variation is relatively slow. An injection of beam into LHC represents an instantaneous (turn-by-turn) increase of the intensity, well above threshold, which doesn't give any noise on the BPF signals.

The noisy transitions for beam intensity close to the threshold were removed for the 2011 implementation [1].

Time Delay

As the individual BPF system channels are independent, a time delay could be observed between the BPF_1A and 1B signals. Table 3 shows an example of this situation. The first two rows have no time difference between flags: they both switch to true at the same time, corresponding to an injection of beam into the LHC. On the other hand, the second and third rows show that the two flags change the state to FALSE at well different times: BPF_1A is delayed by some three minutes compared to BPF_1B:

Table 3: Pick-Up Time Delay

Time	Transition	Flag
09:50:59.410	F→T	BPF_1A
09:50:59.410	F→T	BPF_1B
10:59:21.900	T→F	BPF_1B
11:02:42.541	T→F	BPF_1A
11:30:53.410	F→T	BPF_1A
11:30:53.410	F→T	BPF_1B

Of the 600 transitions analyzed, 93% of them have a delay between flags of less than 3µs. The remaining 7%, with the exception of the single example above, have a delay between 30ms and 90ms. In all cases this corresponded with a very slow gradual decrease in beam intensity, and on a flag transition from TRUE to FALSE. The discrepancy in switching times was attributed to off-centred beam.

2011 IMPLEMENTATION

The characterisation of the BPM implementation in 2010 was very successful, and a revised implementation of the BPF electronics was put into operation in summer 2011. This included four important changes to address observations from 2010 [1]:

Position Dependence

As shown in Fig, 4, signals from the opposite BPM electrodes are summed before being split and passed to the channels of the BPF front-end. This removes most of the dependency on beam position for the evaluation of the flags.

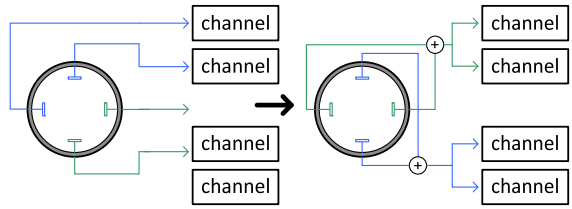


Figure 4: Suppression of Position Dependence.

Hysteresis

Hysteresis was added to remove noisy transitions as the beam intensity passes through the threshold.

Threshold

The threshold of 6×10^8 was considered to be a good demonstration of the system sensitivity, at the same time, was lower that the threshold intensity of the LHC beam orbit measurement system. This system is essential for reliable LHC operation, so beam intensity must guarantee its operation. For that reason the BPF system threshold was raised to about 2×10^9 charges, by attenuating the BPM signals before the BPF analogue front-end (see Fig. 5).

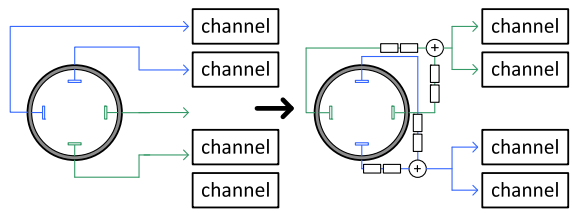


Figure 5: Increase of Threshold.

Fast Beam Current Transformer

Finally, the flags from the FBCT system were disabled from the voting logic of the SMPC. Instead of operating with six BPF and two 2 out of 3 voters, there are four BPF, which must all be TRUE for the corresponding global BPF to be TRUE.

CONCLUSIONS

The prototype beam presence flag system based on BPM signals was tested in principle in 2010 and with minor modifications was put into regular operation for the 2011 LHC run. Whilst the system has been proved to work with a threshold of 6×10^8, it has been implemented with a threshold of 2×10^9 to protect the LHC.

ACKNOWLEDGEMENTS

This work has been the result of a close and productive collaboration between the LHC machine protection and LHC beam instrumentation groups.

REFERENCES

[1] T. Bogey, M. Gasior, "The LHC Beam Presence Flag System", DIPAC 2011, CERN-BE-2011-026.

DEVELOPMENT OF A MACHINE PROTECTION SYSTEM FOR THE SUPERCONDUCTING BEAM TEST FACILITY AT FERMILAB*

A. Warner#, L. Carmichael, M. Church, R. Neswold, FNAL, Batavia, IL 60510, U.S.A

Abstract

Fermilab's Superconducting RF Beam Test Facility currently under construction will produce electron beams capable of damaging the acceleration structures and the beam line vacuum chambers in the event of an aberrant accelerator pulse. The accelerator is being designed with the capability to operate with up to 3000 bunches per macro-pulse, 5Hz repetition rate and 1.5 GeV beam energy. It will be able to sustain an average beam power of 72 KW at the bunch charge of 3.2 nC. Operation at full intensity will deposit enough energy in niobium material to approach the melting point of 2500 °C. In the early phase with only 3 cryomodules installed the facility will be capable of generating electron beam energies of 810 MeV and an average beam power that approaches 40 KW. In either case a robust Machine Protection System (MPS) is required to mitigate effects due to such large damage potentials. This paper will describe the MPS system being developed, the system requirements and the controls issues under consideration.

INTRODUCTION

The beam at Fermilab's New Superconducting RF Beam Test Facility [1], when operational, will need systems to protect critical components from beam induced damages such as beam pipe collision and excessive beam losses. The MPS must therefore identify hazardous conditions and then take the appropriate action before damage is caused. Since the loss of a full bunch train can result in significant damage, the MPS must be able to interrupt the beam within a macro-pulse and keep the number of bunches below the damage potential once the protection system reacts; the goal is to keep the number of bunches on the order of 3-6 bunches. With the high possible bunch frequency of 3 MHz this necessitates a reaction time in the range of 1-2 µs with cable delay included for the 134 metre long machine. The MPS will use the status of critical sub-systems and losses measured by a fast Beam Loss Monitor (BLM) system, using scintillators and photomultiplier tubes (PMT) to identify potential faults. Once a fault is observed, the MPS can then stop or reduce beam intensity by removing the permit from different beam actuators, including the laser pulse controller.

MACHINE LAYOUT

The machine layout is shown in Fig. 1. The accelerator consist of an electron gun, a 40 MeV injector, a beam acceleration section consisting of 3 TTF-type or ILC-type cryomodules initially, and multiple downstream beam lines for testing diagnostics and performing beam experiments. The facility will accommodate up to 6 cryomodules in the final stage. From a machine protection point of view, the dump locations shown describe the final destination of the beam that traverses a path along the beam-line. These paths are termed operation modes and are validated by the MPS before the beam permit is released. The MPS validates these paths by monitoring all critical devices and diagnostics along the path and ensuring that they are all in good status and ready to receive beam at the requested intensity.

The machine will be capable of operating over a wide range of beam parameters as long as the total beam power remains below the limit of the beam dump capability and satisfies radiation shielding requirements. For machine protection purposes several beam modes have been defined; the beam mode sets limits on the number of bunches and therefore the intensity. Initially the following two modes will be active in the system:

- Low intensity mode – which allows the minimal beam intensity needed for OTR/YAG diagnostics. This is below the threshold potential for beam induced damage. In this mode there is no fast reaction to beam loss within a bunch train.
- High intensity mode – which does not impose a limit on the number of bunches, but enables fast intra-train protection by the MPS.

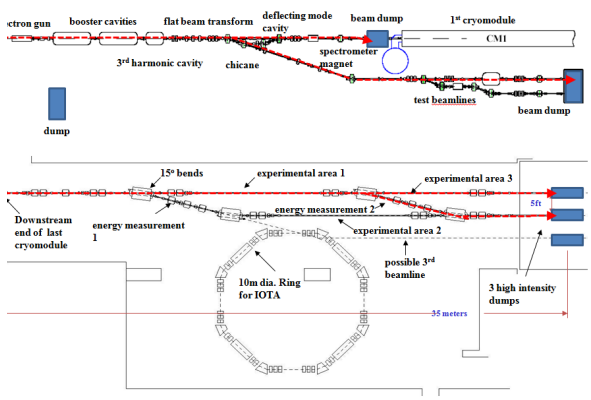


Figure 1: Machine layout.

MPS OVERVIEW

The simplified overview block diagram of the proposed MPS is shown in Figure 2. The MPS has connections to several external devices and sub-systems. The top layer comprises signal providers such as fast beam loss monitors, RF signals, quench protection, toroid transmission, vacuum, magnet power supplies and more. All devices in this category send status information to the MPS logic layer (permit system). Only simple digital

*Operated by Fermi Research Alliance, LLC, under Contract No. DE-AC0207CH11359 with the United States Department of Energy
#warner@fnal.gov

signals (e.g. on-off, OK-not OK) are transmitted. All devices or subsystems that are determined to be pertinent to protecting the machine or necessary for machine configuration are included here. The state of the machine is determined from this comprehensive overview of the inputs and allowable operation modes are determined based on this information by the middle logic layer. The main goals for the MPS system as a whole are:

- Provide precise protection of all critical components by first determining the fault severity (high, intermediate, etc) and then taking the appropriate action to avoid damage.
- Allow for high availability by ensuring that the maximum requested beam intensity is allowed for the detected fault severity.
- Monitor MPS components and perform periodic self-checks in order to ensure robustness and a high level of reliability.
- Provide well-integrated, user-friendly tools for fault visualization, control and post-mortem analysis.

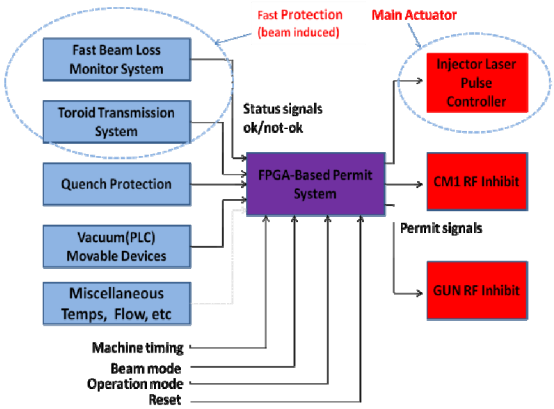


Figure 2: MPS Overview.

The permit system part of the MPS is capable of handling events on all time scales relevant to the machine. This layer is FPGA based and is thus fully programmable and handles complex logic tasks. The logic here will be designed to ensure safe operating conditions by monitoring the status of critical devices and by imposing limits on the beam power. It prohibits beam production or reduces the beam intensity by disabling the gun RF and the injector laser unless the requirements for the specific predefined modes are fulfilled when that mode is requested. The final layer of the system shows the main actuators. This comprises all of the points where the MPS logic may act on the operation of the machine and prevent beam from being produced or transported; the main actuator being the injector laser. When any of the non-masked inputs signal an alarm status the MPS permit system (logic layer) can do one of several things based on the severity of the fault: i.e. switch off the injector laser to suppress the production of new bunches, reduce the intensity by dialing back the number of bunches, or inhibit the rf power from the first cryomodule (CM1) as a precaution against transport of dark current from the RF gun.

BEAM LOSS MONITOR SYSTEM

Several types of beam loss monitors (BLMs) will be used for the detection of electromagnetic showers. The fast protection system is being designed to interrupt the beam within a macro-pulse and will rely heavily on the ability to detect and react to losses within a few nanoseconds; for this reason the primary loss monitors for fast protection are made of plastic scintillator with photomultipliers attached and have already been designed, built and tested. Figure 3 shows some measurement results.

Figure 3: FBLM showing pulse beam losses.

The BLMs will serve the dual purpose as an accelerator diagnostic and as the primary detectors for fast machine protection. The monitors must therefore deliver a measurement of beam and dark current losses to the control system as well as generate a fast alarm signal when the beam losses exceed user-defined thresholds. The time resolution of the loss measurement must provide the ability to distinguish single bunches within each macro pulse. This requires a sampling frequency of at least 3 MHz with a repetition rate of 5 Hz. The BLMs must be integrated into a robust beam loss monitoring system capable of generating an alarm condition that is derived by comparing the outputs of the PMT signals with various programmable thresholds. This alarm output is a critical component for machine protection. The desire is to provide a machine protection trip well before the beam can damage accelerator components. If one of the programmed thresholds is exceeded or if an error condition such as a high voltage failure or failed monitor is detected the system should report this to the MPS logic which in turn reduces the intensity or inhibits the beam.

The main requirements for the BLM system are:

- Provide both machine protection and diagnostic functions.
- Instantaneous read-back of beam loss
- Digital output for integrating and logarithmic signal (16 bit)
- Built in self test and onboard signal injection for testing of monitors between pulses.
- FPGA controlled
- Local data buffer
- VME interface to ACNET control system
- Continuous and pulsed monitoring
- Wide dynamic range

Cryogenic Loss Monitors

Although loss monitors are typically one of the main diagnostics for protecting the accelerator from beam induces damage. Most accelerator facilities do not cover the cold sections of the machine with loss monitors. To address these issues a Cryogenic Loss Monitor (CLM) ionization chamber capable of operation in the cold sections of a cryomodule has been developed and will be installed and tested [2]. The monitor electronics have been optimized to be sensitive to DC losses and the signals from these devices will be used to study and quantify dark current losses in particular, see Figure 4. In order to increase the resolution bandwidth and the response time of the devices a new scheme which uses a Field Programmable Gate Array (FPGA) based Time-to-Digital converter (TDC) method is implemented [3] instead of a standard pulse counting method. This potentially renders these monitors as useful devices for both dark current monitoring and machine protection. These monitors under consideration are custom built detectors. They are an all metal designed which makes them intrinsically radiation hard and suitable for operation at 5 Kelvin to 350 Kelvin.

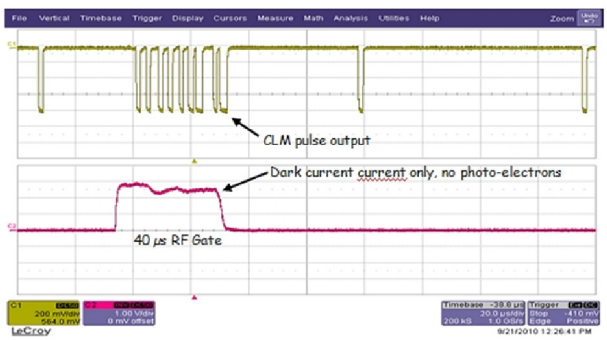


Figure 4: reaction of CLM to dark current losses in a test beam.

LASER PULSE CONTROLLER

One of the main actuators for the MPS is the injector laser pulse control. This will be the device that controls the number and the spacing of bunches in a macro-pulse by picking single laser pulses out of a train. This is achieved by manipulating the Pockels cell (voltage-controlled wave plates). This system is also the main actuator for beam inhibits issued by the MPS. It is envisioned that this would be a VME board with a fully programmable FPGA. It would have inputs for the requested beam modes defined by the logic layer of the MPS, the MPS permit signal, the 3 MHz machine timing, and for a macro-pulse trigger. Based on the laser/gun design, it would have control outputs for the Pockels cell driver, a mechanical shutter and a first bunch timing signal. From the protection system point of view the pulse controller is used to:

- Block the Pockels cell based pulse kickers as long as **any** MPS input is in an alarm state.
- Enforce the limit on the number of bunches as given by the currently selected beam mode.
- Close the laser shutter on request of the MPS. This may happen when there is no valid operation mode or when some combination of loss monitors exceed thresholds which trigger a dump condition.

CONTROLS INTEGRATION

The MPS will need server support for the various hardware systems to View, Configure and Diagnose the system. Already there are currently several servers under development for the beam loss monitor system and the laser pulse controller. These servers were implemented using the PowerPC 5500 series boards running VxWorks 6.4 and implementing the ACNET protocol. Some of the main requirements for these servers include:

- Time-stamping at a sub-microsecond resolution in order to allow for data correlation.
- Circular buffers that are logged using ACNET data loggers and thus provide a repository used for post-mortem analysis.

A control system that is well integrated with all MPS components, from the various front-ends to the high level applications, is critical to leveraging the full functionality of that control system.

CONCLUSION

Significant effort is underway towards developing a reliable MPS for this new facility. System integration and commission challenges lay ahead.

REFERENCES

[1] M. Church, *et al.* “Status and plans for an SRF test facility at Fermilab”, SRF’11, Chicago, August 2011, MOPO006; http://www.JACoW.org.

[2] A. Warner and J. Wu, “Cryogenic loss monitors with FPGA signal processing”, Fermilab preprint FERMILAB-PUB-11-467-AD.

[3] J. Wu, *et al.* “ADC and TDC implemented using FPGA”, IEEE Nuclear Science Symposium and Conference Record 2007;1:281-286.

THE MACHINE PROTECTION SYSTEM FOR THE R&D ENERGY RECOVERY LINAC*

Zeynep Altinbas#, James Jamilkowski, Dmitry Kayran, Roger C. Lee, Brian Oerter,
Brookhaven National Laboratory, Upton, NY 11973, U.S.A.

Abstract

The Machine Protection System (MPS) is a device-safety system that is designed to prevent damage to hardware by generating interlocks, based upon the state of input signals generated by selected sub-systems. It protects all the key machinery in the R&D Project called the Energy Recovery LINAC (ERL) against the high beam current. The MPS is capable of responding to a fault with an interlock signal within several microseconds. The ERL MPS is based on a National Instruments CompactRIO platform, and is programmed by utilizing National Instruments' development environment for a visual programming language. The system also transfers data (interlock status, time of fault, etc.) to the main server. Transferred data is integrated into the pre-existing software architecture which is accessible by the operators. This paper will provide an overview of the hardware used, its configuration and operation, as well as the software written both on the device and the server side.

INTRODUCTION

An ampere class 20 MeV superconducting Energy Recovery Linac (ERL) has been built to test concepts for high-energy electron cooling and electron-ion colliders. One of the goals is to demonstrate an electron beam with high charge per bunch (~5 nC) and extremely low normalized emittance (~5 mm-mrad) at an energy of 20 MeV. The ERL R&D program was started by the Collider Accelerator Department (C-AD) at BNL as an important stepping-stone toward a 10-fold increase of the luminosity of the Relativistic Heavy Ion Collider (RHIC). This program aims to demonstrate CW operation of an ERL with an average beam current in the range of 0.1-1 ampere, combined with very high efficiency of energy recovery [1].

The R&D ERL requires a protection system to prevent damage to hardware due to high beam current. The Machine Protection System (MPS) for the R&D ERL is designed as a device-safety system that generates interlocks based upon the state of input signals generated by select sub-systems. It exists to protect key machinery such as the 50 kW and 1 MW RF Systems. When a fault state occurs, the MPS is capable of responding with an interlock signal within several microseconds. The Machine Protection System inputs are designed to be fail-safe. In addition, all fault conditions are latched and time-stamped.

The ERL MPS is based on a National Instruments hardware platform, and is programmed by utilizing National Instruments' development environment for a visual programming language.

HARDWARE

The MPS runs on a programmable automation controller called CompactRIO (Compact Reconfigurable Input Output). The National Instruments CompactRIO platform is an advanced embedded control and data acquisition system designed for applications that require high performance and reliability. This small sized, rugged system has an open, embedded architecture which allows developers to build custom embedded systems in a short time frame.

The National Instruments CompactRIO device that is used for the MPS is an NI cRIO 9074 (see Fig. 1). The cRIO 9074 is an 8-slot chassis with an integrated real-time processor and an FPGA. The embedded processor is a 400 MHz Freescale MPC5200 that runs the WindRiver VxWorks real-time operating system. The FPGA is a Xilinx Spartan 3 with 2 million gates (46,080 logic cells) and 720 KB embedded RAM. The cRIO 9074 also features a 256 MB nonvolatile memory. CompactRIO combines an embedded real-time processor, a high-performance FPGA and hot-swappable I/O modules to form a complete control system. Each module is connected directly to the FPGA and the FPGA is connected to the real-time processor via a high-speed PCI bus (see Fig. 2).

Figure 1: The National Instruments cRIO 9074 [2].

ERL critical sub-systems such as the RF system require the MPS to respond on a microsecond scale. High-speed I/O modules were chosen to meet the necessary timing requirements. The 24V capable module has sinking digital inputs with 7 µs response time, and the TTL module has digital inputs and outputs with 100 ns response time.

SOFTWARE

The MPS interface is written in LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench).

*Work supported by Brookhaven Science Associates, LLC under Contract Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy.
#altinbas@bnl.gov

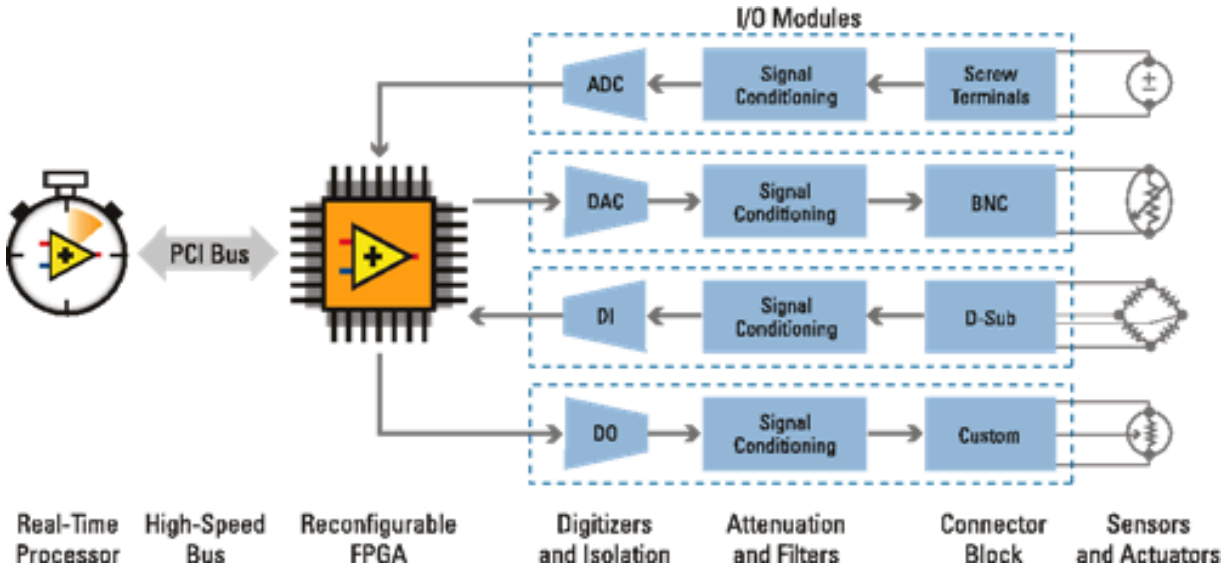


Figure 2: Block diagram showing a generic CompactRIO architecture [2].

LabVIEW is a graphical programming environment used to develop measurement, test, and control systems utilizing graphical icons and wires that resemble a flowchart. The National Instruments CompactRIO platform requires two different LabVIEW software modules corresponding to the System's interface, one for the Real-Time processor and one for the FPGA. These modules contain custom functions specific to the Real-Time processor or the FPGA in addition to all the functionalities of the standard LabVIEW module.

The code for both the Real-Time processor and the FPGA is developed on a host computer. The program for the FPGA is developed by using a standard LabVIEW software module. The LabVIEW FPGA code is then translated to VHDL code and compiled using the Xilinx tool chain. The program for the Real-Time processor is also developed by using a standard LabVIEW software module. When ready, the code for the Real-Time processor and the FPGA is downloaded to the CompactRIO device via Ethernet. Once the code is downloaded, the CompactRIO can run in a stand-alone mode, or communicate directly with a host via Ethernet.

Running directly on the CompactRIO platform, the MPS interface accepts the various input signals and generates any necessary interlocks. An interlock is generated when a logic high (fault) is seen at the input. If a cable is disconnected or broken, an internal pull-up ensures the system will generate an interlock. Exception is given to the RF sub-system which provides normally-high inputs due to equipment constraints. These inputs are inverted within the LabVIEW FPGA. If one of the continuously polled input levels change to high (indicating a fault), the fault is latched, and the time of the event is recorded using a 32-bit LabVIEW tick counter function. This provides a microsecond time stamp. The MPS interface also provides the capability to enable and disable inputs. Enabled and latched inputs are then combined and passed to other critical systems as interlocks. The input latches are cleared only after a software reset has been issued.

Operators communicate to the MPS interface using the main ERL server. Interaction with this Linux server is handled by a separate TCP server using the locally-constructed Simple Network Access Protocol (SNAP). SNAP provides remote access to the statuses of all MPS inputs and enable/disable controls for each. Additional handshake information between the server and the CompactRIO hardware across the network chain is used to monitor the overall status of the system. Commands can be sent and statuses read from the main ERL server process through a standard Collider Accelerator Department (C-AD) text-based parameter editing page that can be accessed from networked workstations or thin-clients. A web server capability is also provided by National Instruments to allow the developer to monitor and control the system remotely, avoiding interaction with the main ERL server.

EXISTING SYSTEM

The original ERL MPS was implemented and tested during the Cold Emission Test in 2009. In this configuration, the MPS had nine sub-system inputs and four outputs. As per the requirements, each sub-system provides either a TTL level or normally-open dry contact. All the inputs are latched when a fault occurs, and the fault time is recorded. This information is displayed by the MPS interface until the reset button is activated in an attempt to clear latched inputs. If the fault condition is still present, the appropriate latch maintains the interlock status. When the MPS turns on initially, the same reset button is used to establish a connection and inform the hardware that the sub-systems are ready to be monitored. This practice is utilized in order to prevent the MPS from starting prematurely in the case of a power-loss. The interface also has an indicator derived from the inputs, to

display the status of the critical sub-systems. The system response time is ~3-4 µs.

The MPS communicates with the ERL main server by sending data every 10 minutes, or every time a value changes.

PERFORMANCE

The MPS for the R&D ERL was recently used during the conditioning of the fundamental power couplers (FPC) [3]. The existing CompactRIO chassis is populated with 4 I/O modules: a 32-channel 24V capable input module, two 8-channel TTL input-output modules, and a 4-channel SPST relay output module. The layout is illustrated in Fig. 3. The inputs were expanded from nine to twenty-eight for this test. This did not require any additional I/O modules. The response time has increased to ~8 µs with the addition of the necessary logic and the defined I/O channels. This increase occurs because once more than 8 channels per module are utilized CompactRIO transitions to a byte-wide polling scheme. In other words, for up to 8 channels, the I/O is routed directly to the FPGA; when more than 8 channels are used, the I/O is transferred using a National Instruments proprietary Serial Peripheral Interface (SPI). The MPS hardware can be expanded by installing more I/O modules into the 4 remaining available slots in the CompactRIO chassis. The overall system size can also be expanded by daisy-chaining to another chassis via EtherCAT [4].

The Commercial of the Shelf (COTS) hardware-based MPS is extremely time and cost effective compared to a custom design. The initial configuration of the system took about three months and modifications for the FPC conditioning took about a week. This short development time compliments the system flexibility and expandibility, resulting in a very effective solution with minimal resource requirement. The whole system, including the software modules, cost about US ~$5,000.

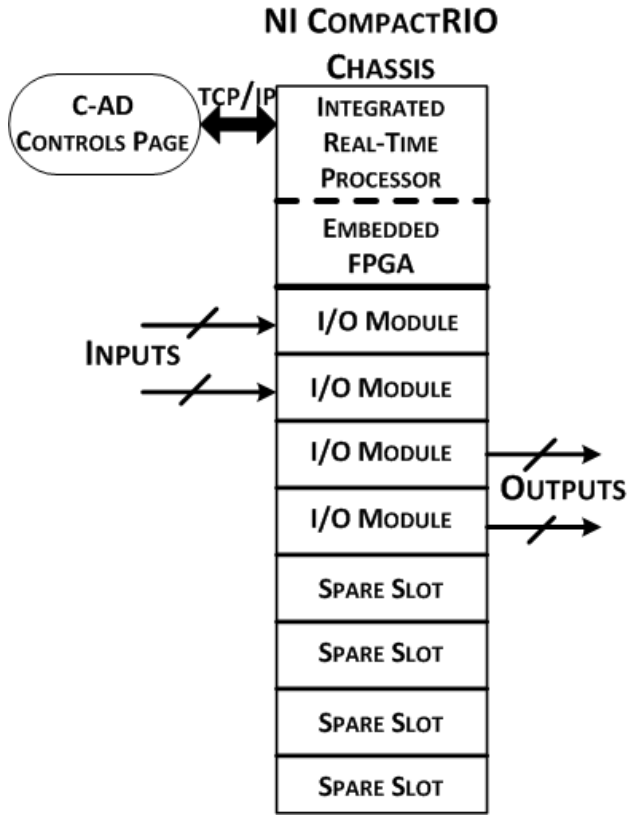


Figure 3: Block diagram for the current layout.

REFERENCES

[1] D. Kayran et al., "Status of High Current R&D Energy Recovery Linac at Brookhaven National Laboratory", PAC'11, New York, March 2011.
[2] Image is courtesy of National Instruments.
[3] W. Xu et al., "FPC Conditioning Cart at BNL", PAC'11, New York, March 2011.
[4] http://www.ethercat.org/default.htm Accessed: 10/03/2011.

PRE-OPERATION, DURING OPERATION AND POST-OPERATIONAL VERIFICATION OF PROTECTION SYSTEMS

Iván Romera¸ Maxime Audrain, CERN, Geneva, Switzerland

Abstract

This paper will provide an overview of the software checks performed on the Beam Interlock System ensuring that the system is functioning to specification. Critical protection functions are implemented in hardware, at the same time software tools play an important role in guaranteeing the correct configuration and operation of the system during all phases of operation. This paper will describe tests carried out pre-, during- and post-operation, if protection system integrity is not sure, subsequent injections of beam into the LHC will be inhibited.

INTRODUCTION

In order to prevent damage in the Large Hadron Collider (LHC) due to uncontrolled beam losses or failures, a Machine Protection System has been implemented which detects potential dangerous situations and eventually extracts the circulating proton beam of the machine into a graphite absorber. The heart of this system is the Beam Interlock System (BIS) [1] which allows a fast and reliable transmission of beam dump requests to the LHC Beam Dumping System (LBDS).

The BIS has been designed as a highly dependable system, meaning that a failure of the BIS could lead to a sequence of events that result in a significant damage of the LHC. To prevent this and in order to verify the integrity of the BIS, a set of operational checks have been implemented.

MOTIVATION

To protect the LHC ring from beam damage the BIS relies on 17 Beam Interlock Controllers (BIC) installed in the underground areas spanning over more than 27 km. The BICs are linked using redundant optical fibres which define the Beam Permit Loops (BPL). These loops propagate beam dump requests, from more than 190 user systems connected to the BIS, to the LHC Beam Dumping System (LBDS). With such a complex distributed system, the need of having software operational tools for diagnostics is a must.

The aim of the operational tools is to verify the system´s operation from end-to-end and to inhibit next beam injection into the LHC in case of a problem detected.

The scope of the project covers all software modules implemented to provide to the operators and equipment experts the necessary tools to guarantee the safe operation and to ensure that the system is in an “as good as new” state.

The verification of the BIS is carried out in three stages: (1) pre-; (2) during- and (3) post- operational checks which are executed on demand by the LHC Sequencer [2] depending on the operational mode of the machine. The outcome of the checks leads to the actions represented in Table 1.

Table 1: Result of Operational Checks

Test	Passed	Failed
Pre-	Injection allowed	Injection inhibited
During-	Continue operation	Warning experts
Post-	Injection allowed	Injection inhibited

The operational tools described hereafter are implemented with the standard tools and infrastructure used in the CERN accelerator control system (i.e.: Java, Swing, Spring Framework, Controls Middleware CMW and the Controls Configuration Database CCDB).

PRE-OPERATIONAL CHECKS

Before injecting beam in the machine, it is required to ensure that all safety critical components of the BIS are correctly configured and ready for operation. The configuration of such relevant components is defined in an Oracle reference database (CCDB) and maintained by the system experts with a strict versioning and access control. A wrong configuration of a critical part of the BIS could have serious implications on safety (e.g.: in case of a user system input that should never be masked is connected to a maskable input); it could lead to miss a beam dump request from a user system and in the worst case could lead to important damage to the machine.

Pre-Operational Checks Implementation

The pre-operational checks have been implemented as a sequence in the middle tier of the LHC Sequencer, which is a high level software application in charge of orchestrating the execution of tasks in a controlled way, helping the operators to drive the LHC. The pre-operational tests are launched programmatically before the beam injection phase, guaranteeing the consistency in the configuration of the physical system against the reference database.

In case of encountering any inconsistency, a failure will be propagated to the BIS by disabling the software permit through the Software Interlock System (SIS) which will inhibit next beam injection (see Fig. 1). In addition, a new entry on the LHC logbook will be created for log tracking.

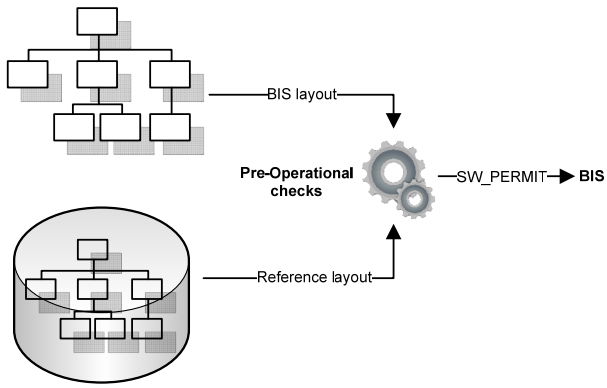


Figure 1: Pre-Operational checks workflow.

The sequence of checks implemented includes:

- Jumper configuration: it guarantees that no masked hardware inputs exist. It verifies that both redundant input channels from the user systems are enabled if required for operation.
- User system presence: it verifies the existence of the user system interfaces on those inputs defined as enabled in the database.
- User system ID: it prevents crossings between user system connections. It verifies that the unique identifier of the user system interface matches with the one defined in the database.

Besides, it is possible to perform on demand tests with each individual user system of the BIS. These tests consist in requesting the user system to force independently both redundant channels to the BIS to a false state to guarantee that the system is able to perform a beam dump request. Due to timing constraints this action cannot be performed on every user system before injection, however it is strongly recommended to be executed on a regular basis (e.g.: technical stops) to prevent possible blind failures of the user system interfaces.

DURING-OPERATION CHECKS

During beam operation, it is required to verify the integrity of the infrastructure and that several critical elements behave correctly and within tolerances. The online monitoring can have some impact on the dependability of the system, as despite critical protection functions are built into the hardware, the online verification can contribute to improve the availability of the system (e.g.: detection of a faulty redundant power supply that can be repaired during a technical stop in the shadow of beam operation).

In case of errors detected, the equipment experts will be informed but no other actions will be taken.

Implementation in DIAMON

The during- operation checks have been implemented as monitoring agents in the DIAMON [3] framework, which is a monitoring tool in charge of diagnosing problems in the controls infrastructure. This solution was chosen as it is a standard and flexible tool used in the controls group that provides embedded services such as access to the logging system or role-based access control to restrict the access to the agents. The checks are structured into two parts: (1) related to the infrastructure and (2) related to the critical hardware itself. With regard to the infrastructure the agents check:

- Frontend VME hardware (redundant power supply status, fan speed, fan temperature, …).
- VME Operating system parameters (memory and CPU load, disk space, system resources, user processes running …).

With respect to the critical hardware, the agent monitors:

- pulse per second reception and timing alignment problems from the LHC Machine General Timing (GMT).
- spurious glitches in the reception of critical fail-safe signals (i.e.: high radiation levels can cause glitches on the link. See Fig. 2).
- matrix clock frequency within predefined tolerances.
- redundancy of power supplies for user interfaces with the user systems.
- consistency of non-critical serial channels used for monitoring.

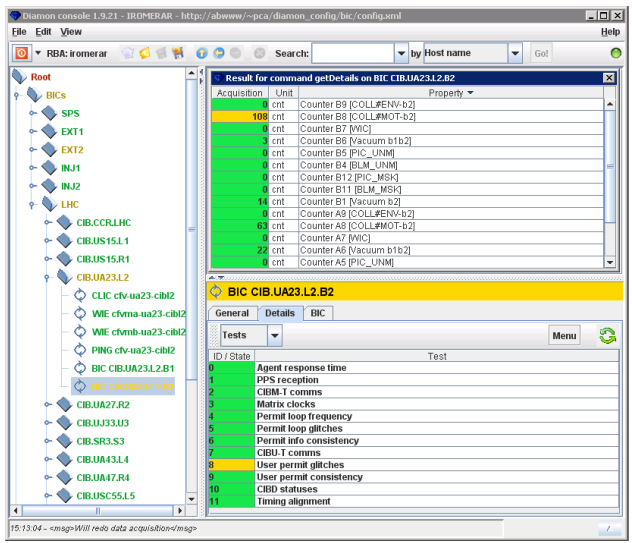


Figure 2: BIC agent in DIAMON console.

POST-OPERATIONAL CHECKS

After a beam dump, it is required to identify that the BIS was not the source of the beam abort and that all safety critical components of the system worked as expected. If the protection system integrity is compromised, beam operation must be inhibited until the problem is solved.

The post- operational checks were designed to fulfil the following requirements:

- be triggered after the reception of a PM event

- collect buffers from every BIC in the LHC and Injection Lines (INJ) as well as from LBDS system to reconstruct the sequence of events
- receive configuration data from reference database
- perform system checks and compare measured data against references
- inhibit beam operation via a software channel to the BIS if any relevant problem is found during the analysis
- display a summary of the checks performed, providing clear information of problems identified
- provide analysis of archived dumps

Furthermore, the post- operational checks of the BIS have to provide clear information about who was the first protection system provoking the beam dump.

Implementation in the Post-Mortem Framework

The Internal Post Operational Checks (IPOC) have been implemented as an analysis package in the PM framework [4], which is the system in charge of identifying the cause of the beam abort by performing individual system checks on the different equipment systems. These checks are immediately launched after an emergency dump. In such cases a PM event will be distributed all around the machine via the GMT and the PM buffers of the BICs are frozen, covering a time window around the event. The data PM buffers are pushed to the LHC PM server through a dedicated client library for analysis.

The IPOC module covers the following aspects of the BIS (e.g.: Fig. 3 shows the IPOC module being part of the PM framework):

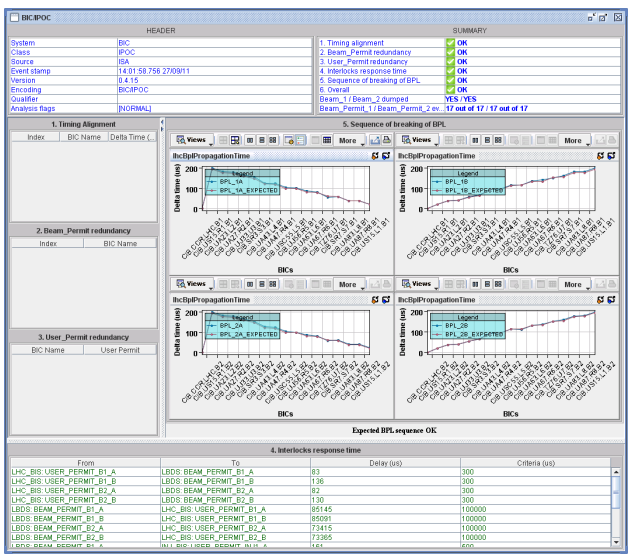


Figure 3: BIS-IPOC console.

- all records in the PM buffers of all BICs are time-stamped with the same timing references. This is vital for correlation of data.
- redundancy in the communication channels from the user systems up to reception of dump requests to LBDS is working.
- optical propagation delays in the Beam Permit Loops are as expected when comparing to the references.
- propagation of interlock signals between BIS and LBDS are correct and within tolerance.

CONCLUSIONS AND FUTURE PLANS

Operational verification tools must guarantee the correct configuration and integrity of the critical components of the BIS during all phases of beam operation. They have been intensively used during more than two years of LHC operation and partially deployed on the injector complex. It is demonstrated that the use of such tools plays an important role on prevention and diagnosis of problems and allowed gaining confidence in the reliability of the BIS.

In the near future it is foreseen to consolidate and improve the existing tools and to extend the scope of these checks to the BIS of the Super Proton Synchrotron (SPS) and related extraction areas.

REFERENCES

[1] B.Todd et al. “The architecture, design and realisation of the LHC Beam Interlock System”, ICALEPS’05, Geneva, Switzerland, October 2005

[2] V. Baggiolini et al., “A Sequencer for the LHC era”, ICALPECS’09, Kobe, Japan, October 2009

[3] J. Lauener et al., “The DIAMON Project – Monitoring and diagnostics for the CERN Controls Infrastructure”, EPAC’04, Lucerne, Switzerland, July 2004

[4] M. Zerlauth et al., “The LHC Post Mortem Analysis Framework”, ICALEPCS’09, Kobe, Japan, October 2009

SAFETY CONTROL SYSTEM AND ITS INTERFACE TO EPICS FOR THE OFF-LINE FRONT END OF THE SPES PROJECT

J. Vasquez*, A. Andrighetto, G. Bassato, L. Costa, M. Giacchini, INFN, Legnaro, Italy
M. Bertocco, UNIPD, Padova, Italy

Abstract

The SPES off-line front-end apparatus involves a number of subsystems and procedures that are potentially dangerous both for human operators and for the equipments. The high voltage power supply, the ion source complex power supplies, the target chamber handling systems and the laser source are some example of these subsystems. For that reason, a safety control system has been developed. It is based on Schneider Electrics Preventa family safety modules that control the power supply of critical subsystems in combination with safety detectors that monitor critical variables. A Programmable Logic Controller (PLC), model BMXP342020 from the Schneider Electrics Modicon M340 family, is used for monitoring the status of the system as well as controlling the sequence of some operations in automatic way. A touch screen, model XBTGT5330 from the Schneider Electrics Magelis family, is used as Human Machine Interface (HMI) and communicates with the PLC using MODBUS-TCP. Additionally, an interface to the EPICS control network was developed using a home-made MODBUS-TCP EPICS driver in order to integrate it to the control system of the Front End as well as present the status of the system to the users on the main control panel.

INTRODUCTION

Selective Production of Exotic Species (SPES) is an Istituto Nazionale di Fisica Nucleare (INFN) Project for the development of the Nuclear Physics as an intermediate step toward EURISOL and to cover interdisciplinary applied physics in the fields of material science and medical applications. The project is based on a facility (Fig. 1) for the production of neutron-reach Radioactive Ion Beams (RIB) using the Isotope Separation On-Line (ISOL) technique [1, 2].

The radioactive ion beams are produced by proton induced fission on a Uranium Carbide (UCx) direct target at a rate of $10^{13} fission/s$. The radioactive ions are re-accelerated at energies higher than 10 $AMeV$ for mass region $A = 130$. Re-acceleration will be performed by the superconducting linear accelerator complex (PIAVE-ALPI) of the LNL. The expected beam-on-target is on the order of 10^8 pps for ^{132}Sn, ^{90}Kr, and about $10^5 - 10^6$ pps for ^{134}Sn, ^{95}Kr considering a total efficiency of 2% from the $+1$ source to the experimental target [2, 3, 4, 5].

* jesus.vasquez@lnl.infn.it

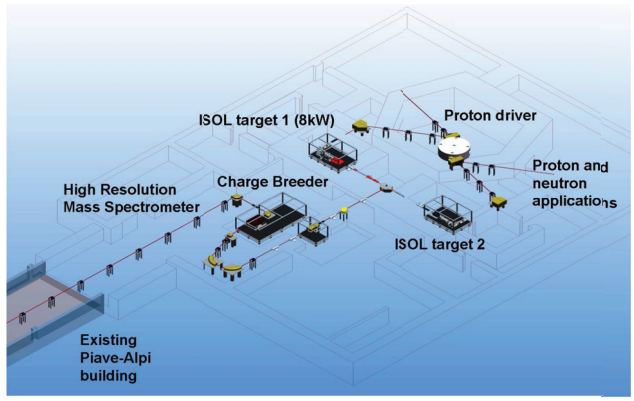


Figure 1: The SPES facility.

At the present, a test bench, called the Front End (Fig. 2), has been constructed at the LNL. It is an apparatus for the acceleration of stable +1 ions up to energies of 30 keV. It permits testing and improving the key systems used on the RIB production like the target complex, the ionization devices and the beam focalization and transport subsystems.

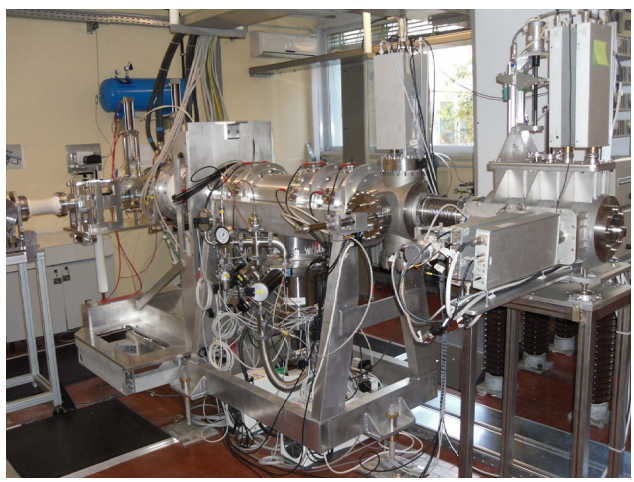


Figure 2: The off-line Front End of the SPES project.

The control system of the Front End has been developed using heterogeneous hardware and software solutions but they are integrated under an unique communication model based on the "Channel Access" of the set of open source tools EPICS (Experimental Physics and Industrial Control System) [6]. Moreover, thanks to this unification, the entire apparatus can be controlled from a main control panel developed using Control System Studio (CSS) collection of user interface tools.

The SPES off-line Front End apparatus involves a number of subsystems and procedures that are potentially dangerous for both human operators and equipments. Among the most potentially dangerous systems are: the high voltage power supply, the ion source complex power supplies, the target chamber handling mechanisms and the laser source.

In order to prevent possible injuries to the operators and damages to the equipments, a safety interlock system has been developed. Furthermore, this system has been interconnected to the EPICS Channel Access in order to have and unify control system.

THE SAFETY CONTROL SYSTEM

The Safety control system have been designed using self-controlled devices and applying redundancy for achieving a PL e/Cat. 4 (EN/ISO 13849-1) and a SIL3 (EN/IEC 62061) safety level. It was developed as a decentralized, distributed structure using Schneider Preventa safety devices. This safety modules are in charge of enabling or disabling the systems that are potentially dangerous, according to the conditions of the system.

The enabling or disabling of a system is done by the safety device controlling two contactors, serially connected, that transmit the power supply to the system. Controlling two contactors increase the reliability of the protection.

The conditions are determined by the status of critical variables (from the safety point of view) that are continuously being monitoring by the same safety devices. To maintain a required level of reliability, the monitoring of the variables is carried out using safety devices.

The critical variables monitored by the safety devices are: the status of the isolation cage gates (using safety magnetic limit switch), the presence of a person around the target chamber (using safety mats), the activation of the command to move the target chamber (using a safety two-hand controller), the activation of an emergency stop (using safety emergency push buttons), the temperature of the target chamber surface (using thermocouples and PT100 devices), the air system pressure (using a pressure switch), the water system flux (using flux meters) and the vacuum level inside the Front End (using vacuum heads and a pressure switch).

A system can be controlled by as much safety devices as critical variables are associated to its operation. Each pair of contactors (for each safety device) is serially connected to the others. In this way, only when every one of the safety devices enables the system, the power supply is transmitted and the system is power on.

For each system, an additional contactor has been added. It is controlled directly by a PLC in order to be able to power on the system when the user requires it, once the right conditions are satisfied. As an additional function, the PLC monitors the status of all the safety modules. A touch screen is used to allow the user to power on and off each system (when possible) and to show him the status of the system.

The PLC used is a BMXP342020 from the Schneider Electrics Modicon M340 family, while the touch screen is a XBTGT5330 from the Schneider Electrics Magelis family. They communication is carry out through the MODBUS-TCP protocol.

The system controlled by the safety controls are: the high voltage power supply, the ion source complex power supplies, the target chamber handling systems and the laser source. Figure 3 shows the safety system for the laser source as an example.

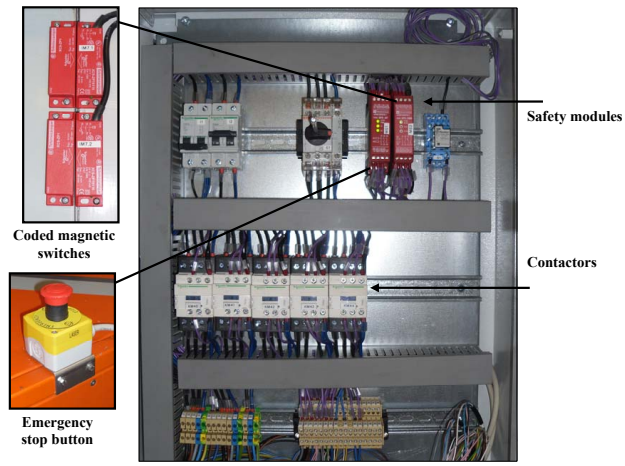


Figure 3: The LASER safety control system.

THE INTERFACE TO EPICS

An EPICS IOC (Input Output Controller) implementing a home-made MODBUS-TCP driver was developed in order to interconnect the EPICS and the safety control systems. This driver requests the PLC the status of the variable associated to the safety control systems and write them on EPICS records, available thus to the entire system through the EPICS Channel Access (CA).

The IOC is implemented on a Linux PC using two Ethernet interfaces: one is use for the PLC communication (MODBUS-TCP) and the other one is use for the EPICS communication (Channel Access). (Fig. 4). On the other hand, the MODBUS-TCP driver was developed using the "StreamDevice" device support for EPICS.

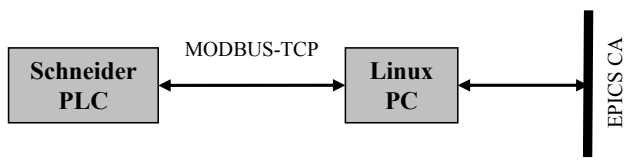


Figure 4: EPICS and safety control systems interconnection.

The MODBUS request is assembled on the IOC record data base and then it is encapsulated on the TCP/IP datagram by the StreamDevice device. The resulting packet is send to the PLC through the corresponding Ethernet interface.

The PLC receives the request, processes it, and them sends the respond through the same Ethernet interface to the IOC. This packet is first received by the StreamDevice who extracts the MODBUS response from the TCP/IP datagram. Next, this MODBUS response is delivered to the IOC record data base where the requested information (the variable status) is read, process and placed on suitable records that are accessible on the Channel Access.

Once the variable statuses are accessible on the Channel Access, they can be monitored by any Channel Access client inside the EPICS network. In our particular case, the statuses of the variables are presented to the user on the Front End main control panel (Fig. 5).

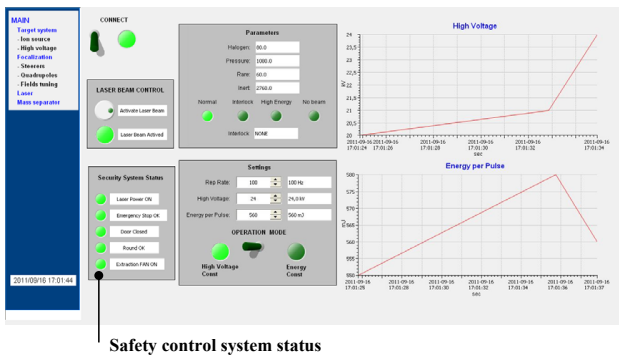


Figure 5: Safety control systems status presented on the LASER control GUI.

CONCLUSIONS

The SPES off-line Front End apparatus involves the use of potentially dangerous equipments. Additionally, the project is getting close to its operative time. Consequently, it was necessary to develop and implement a safety control system to protect both, the equipments and, more importantly, human operators that will be involve soon with the system.

The fact that the project involves dangerous conditions to human being, it demands that the safety control system must be as reliable as possible. Because of that, a distributed hardware-oriented safety control structure was implemented, using special safety devices and techniques achieving PL e/Cat. 4 (EN/ISO 13849-1) and a SIL3 (EN/IEC 62061) safety levels.

Furthermore, although the safety control system is decentralized and not controlled by any other system, a surveillance PLC collects the status information and presents it to the user locally on a touch screen and remotely on the Front End main control panel through an interface to the EPICS Channel Access.

REFERENCES

[1] A. Andrighetto, L. Biasetto, M. Manzolaro, P. Benetti, S. Carturan, P. Colombo, F. Gramegna, G. Meneghetti, B. Monelli, G. Prete, and P. Zanonatoe. The SPES Project at LNL. volume 1099, pages 728–732, March 2009.

[2] P. Zanonato, D. Zafiropoulos, M. Tonezzer, V. Rizzi, G. Prete, L. Piga, C. Petrovich, G. Meneghetti, G. Maggioni, F. Gramegna, P. Di Bernardo, A. Dainelli, P. Colombo, M. Cinausero, S. Cevolani, F. Cervellera, S. Carturan, M. Barbui, C. Antonucci, and A. Andrighetto. The SPES direct UCx target. *Eur.Phys.J. ST*, 150:273, November 2007.

[3] A. Andrighetto, L. Biasetto, M. Manzolaro, D. Scarpa, J. Montano, J. Stanescu, P. Benetti, I. Cristofolini, M. S. Carturan, P. Colombo, P. di Bernardo, M. Guerzoni, G. Meneghetti, B. Monelli, G. Prete, G. Puglierin, A. Tomaselli, and P. Zanonato. Production of high-intensity RIB at SPES. *Nuclear Physics A*, 834(1-4):754c–757c, March 2010. The 10th International Conference on Nucleus-Nucleus Collisions (NN2009).

[4] A. Andrighetto, S. Cevolani, and C. Petrovich. Fission fragment production from uranium carbide disc targets. *Eur.Phys.J. A*, 25:41–47, 2005.

[5] G. de Angelis, A. Andrighetto, G. Bassato, L. Biasetto, L. Calabretta, M. Comunian, A. Galata, M. Giacchini, F. Gramegna, A. Lombardi, M. Manzolaro, G. Prete, L. Sarchiapone, D. Scarpa, D. Zafiropoulos, and the SPES collaboration. Future Perspectives of the Legnaro National Laboratories: The SPES project. *Journal of Physics: Conference Series*, 267(1):012003, 2011.

[6] M. Giacchini, A. Andrighetto, G. Bassato, L. Costa, R. Izsak, G. Prete, and J. Vasquez. The control system of spes target: Current status and perspective. pages 278–280, 2009.

REAL-TIME PROTECTION OF THE "ITER-LIKE WALL AT JET"

M. Jouve[1], C. Balorin[1], D. Kinna[2],
G. Arnoux[2], P. Carvalho[3], S. Devaux[4], P. Thomas[2], K-D. Zastrow[2],
J. Veyret[5] and JET EFDA Contributors*
JET/EFDA Culham Science Center, Abingdon, OX14 3DB, UK
[1]CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France
[2]Euratom/CCFE Fusion Association, Culham Science Center, Abingdon, OX14 3DB, UK
[3]Associação Euratom/IST Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico 1049-001 Lisboa, Portugal
[4]Max-Planck-Insitut für Plasmaphysik, EURATOM-Assoziation, D-85748 Garching, Germany
[5]Sundance France, 22550 Matignon, France

Abstract

During the last JET tokamak shutdown a new ITER-Like Wall [1] has been installed using Tungsten and Beryllium materials. To ensure plasma facing component (PFC) integrity, the real-time protection of the wall has been upgraded through the project "Protection for the ITER-like Wall" (PIW). 13 CCD robust analogue cameras view the main areas of plasma wall interaction and regions of interest (ROI) are used for monitoring in real time the surface temperature of the PFCs. For each camera, ROIs will be set up pre-pulse and, during plasma operation, surface temperatures from these ROIs will be sent to the real time processing system for monitoring and preventing damages on PFCs by modifying the plasma parameters. Since 2005 a real time safety system for the monitoring of the PFCs is routinely used on the tokamak Tore Supra [2]. Based on this successful experience, a similar video and associated control system has been implemented at JET for the PIW. The overall system and the first results are presented in this paper.

THE PIW PROTECTION CAMERAS

The main criteria for selecting the PIW protection camera were guided by the JET hostile environment and especially:

- the robustness under high magnetic field
- the resistance to neutrons and radiations.

The chosen HITACHI black and white CCD analogue camera was used successfully in the past on JET. The simplicity of its technology makes this camera very robust under magnetic field.

The main characteristics and used settings of this camera are:

- Sensor: 752x576 pixels
- Output: 752x288 analogue Video non-interlaced
- 50 fields per second
- External synchronisation
- Exposure time: 20ms
- Standard IR cut filter removed and replaced with a near IR narrow band pass filter centered on 1µm with 50 nm width.

The expected dynamic range is shown is shown on Figure 1 with various optical apertures.

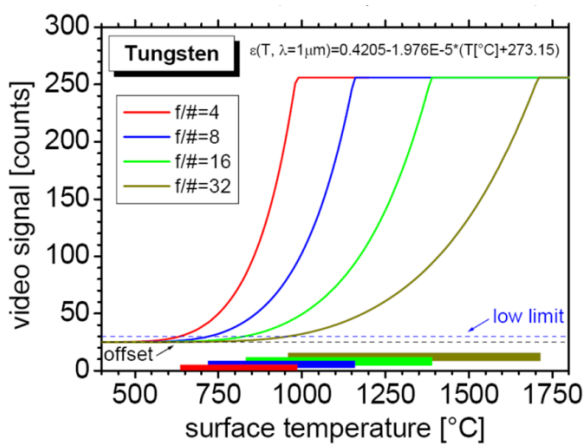


Figure 1: Hitachi camera dynamic range NIR filter : 1µm, Δλ : 50 nm.

VIDEO DIGITIZATION AND DISTRIBUTION

The video is captured using a PLEORA iPort PT1000 frame grabber and it is sent on GigE network through an optical fiber link.

The main characteristics of the iPort are:

- Analogue video input
- GigE Ethernet output
- Sample rate: 15 MHz
- Multicast capability

The iPort multicast capability allows the video distribution through a network switch to 3 different systems as shown in Figure 2.

- Real Time Processing System for PIW protection
- Video capture and replay system for data storage
- Live display system for visualization in JET control room

*See the Appendix of F. Romanelli et al., Proceedings of the 23rd IAEA Fusion Energy Conference 2010, Daejeon, Korea

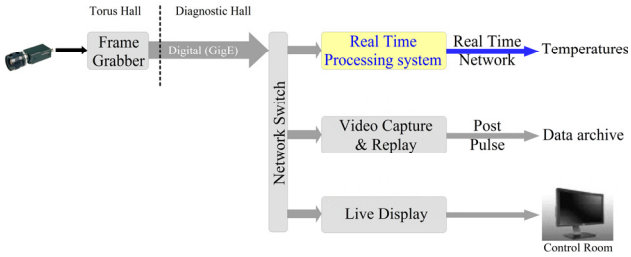


Figure 2: Video digitization and distribution synoptic.

THE REAL TIME PROCESSING SYSTEM

The Real Time Processing System (RTPS) has been divided into a 'Real Time Processing Unit' (RTPU), for surface temperature calculation, and a 'RTPU Host', for connection between RTPU and other systems.

The RTPU host is based on a standard industrial PC with Windows Embedded operating system.

As shown in Figure 3, the choice have been made to manage 2 RTPU with one JET standard industrial PC so one standard PC will manage 2 cameras.

This choice will reduce the risks by distributing the processing units in different host computers. Then for the 13 cameras that will be installed, 7 PCs are necessary.

The main functions of the RTPS are:

- Capture the camera data from Pleora iPORT modules with Intel Pro 1000 network boards and send them to RTPUs through the PCIe bus
- Get all the configuration files and values from JET Database and "Level One" then send them to the RTPUs through the PCIe bus. The configuration data are the ROI map, the dead pixel map, the Non Uniformity Correction matrix (NUC: offset and gain correction) and the calibration files that are Lookup Table (LUT) with 6 possible LUTs so 6 different materials and/or optical transmission can be managed (Beryllium, Tungsten, Tungsten coating…)
- Collect results (maximum temperature values for each ROI…) from RTPUs and send them through the JET Real Time Data Network to the Vessel Thermal Map (VTM) [3]
- Handle the central timing for synchronisation with JET pulses
- Allow ROI display for proper positioning check

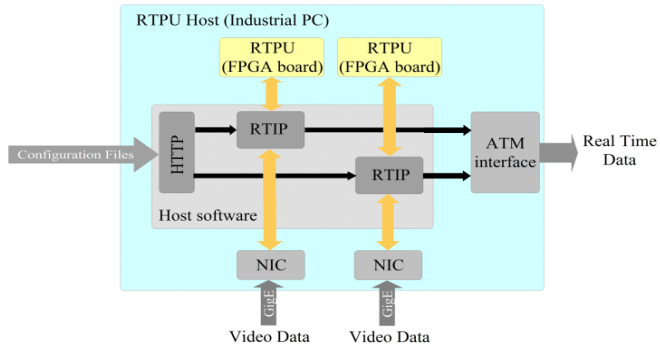


Figure 3: RTPS synoptic.

THE REAL TIME UNIT

The RTPU design, shown on Figure 4, is based on commercial Xilinx Virtex5 FPGA boards (Field Programmable Gate Array).

The on board FPGA ensures the PCIe bus and the DDR2 RAM management and of course the PIW protection functionalities.

All the parameters issued from configuration files are loaded in the on board DDR2 RAM before JET pulses. Using Video Frame Buffer Controller (VFBC) and FIFO, the video and parameters are sent synchronously to the PIW protection engine.

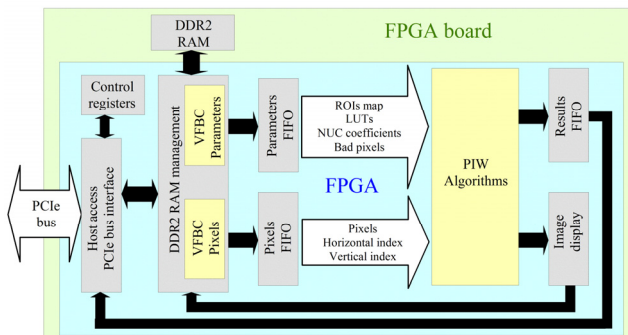


Figure 4: RTPU synoptic.

Programmed under Simulink using Xilinx System Generator block set, the FPGA can manage simultaneously up to 96 ROI per camera which can have any shape.

Each ROI is defined pixel by pixel and can be split in several different areas and can overlap with others. Each pixel can belong to up to 6 ROIs.

The main functionalities included in the RTPU FPGA algorithms (schematically shown on Figure 5), are:

- The PCI express bus and the on board DDR2 RAM management
- Two input FIFO interfaces for synchronisation between algorithms and RAM that are the "Image input" FIFO which handles the pixels value, vertical and horizontal synchronisation and the "Parameters" FIFO which handles the ROI and the Dead pixels maps, the NUC matrix and the used LUT for each pixel

- One "Results" output FIFO which handles all the results for each ROI
- One "Image processed" output FIFO which is mandatory for ROI map control
- The specific PIW algorithms are:
  - Dead pixels and NUC corrections
  - Neutrons impacts removal using configurable filter (Salt and Pepper 3x3 or Median 3x3)
  - Apparent surface temperature calculation using LUT (Lock Up Table) with 1 LUT per ROI chosen among the 6 stored in FPGA memory
  - For each ROI the following values are calculated : Hottest pixel surface temperature (T), hottest pixel position (x,y coordinates), uncertainty estimation using a dynamic sub ROI algorithm centered on hottest pixel (ie: Count number of pixel N in the sub ROI with value greater than X % of T, Hottest pixel is valid if N > threshold, N and X are configurable)
- And control registers which handles the filter selection, the dynamic sub ROI setup, the ROI highlighting…

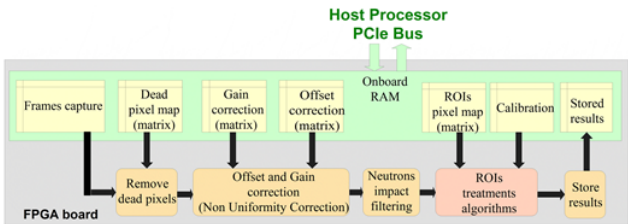


Figure 5: FPGA embedded algorithms.

THE USED COMMERCIAL FPGA BOARD

The chosen commercial board is a "Sundance Multiprocessor" PCI express SMT122T FX70T board, schematically described on Figure 6, for which the main features are:

- 4 lane PCI express interface
- 1 FPGA Xilinx Virtex5 FX70T FGG665
- 2 DDR2 memory banks (256Mbytes per bank)
- 1 Serial PROM 64Mbit
- 2 Marvell 10/100/1000 Ethernet PHYs

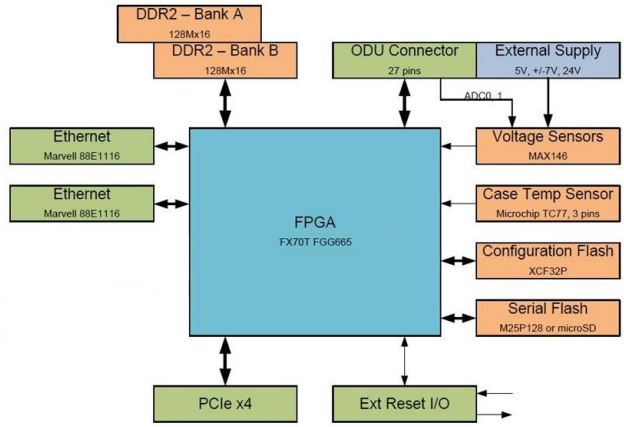


Figure 6: SMT122T FX70T synoptic.

TOOL FOR ROI DEFINITION

A graphical tool, running under Matlab, has been designed for creating the ROI definition file.

This tool allows the ROI drawing using JET background images. The main functionalities are:

- ROI drawing and naming with multiple sub-ROI capability
- ROI or sub-ROI deleting
- Area exclusion inside ROI
- Material (LUT) to ROI assignment
- RTPS readable file creating

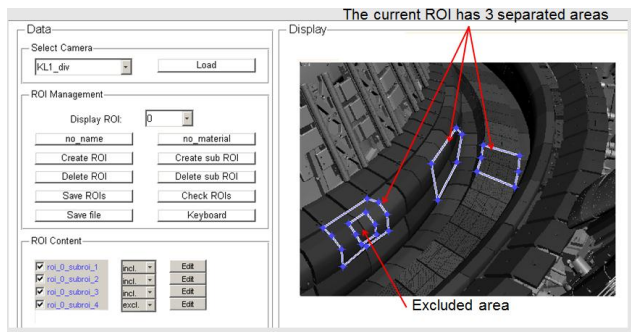


Figure 7: Software interface view.

EXAMPLE OF OBTAINED RESULTS

A preliminary result obtained using image processed displayed and recorded movies playing capabilities has been obtained and is shown on Figure 8.

In this example, 6 ROIs have been used, the results are displayed in the bottom of the image:

- Maximum ROI value
- Maximum coordinate (y,x)
- ROI number of pixel (used for ROI integrity verification)
- Number of pixel in the sub ROI with value
  - greater than X % of hottest pixel value (used for uncertainty evaluation)

The ROI number 0 is highlighted for control.

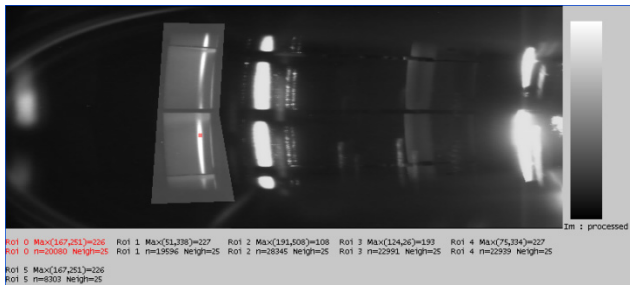


Figure 8: Obtained results example.

CONCLUSION

In the frame of the JET's ITER Like Wall project (first wall in Beryllium and divertor in Tungsten), a real time protection system for these plasma facing components (PIW) has been installed during last shutdown. Composed with a set of 13 CCD cameras, the protection system is based on one commercial FPGA board per camera. In the proposed approach, each FPGA board can manage up to 96 ROIs whilst for each ROI the value and position of the hottest pixel are delivered and sent to JET's VTM. The system has been successfully commissioned and validated during JET's experiments and plasma operations have been stopped by PIW system following a detection of pixels above the threshold limit. The system is now ready for routine operation and it will be particularly used for further development of high performance plasma scenario.

REFERENCES

[1] Guy Matthews et. al. Overview of the ITER-like wall project, Physica Scripta Topical Issue, T128 (2007), 137 – 143.

[2] Philippe Moreau et al. "RF heating optimization on Tore Supra using feedback control of infrared measurements", Fusion Engineering and Design, 2007, Vol 82, 1030-1035.

[3] Adam Stephen et al. "Centralised Coordinated Control To Protect The JET ITER-like Wall.", ICALEPCS 2011 - 13th International Conference on Accelerator and Large Experimental Physics Control Systems.

FIRST OPERATIONAL EXPERIENCE WITH THE LHC BEAM DUMP TRIGGER SYNCHRONISATION UNIT

A. Antoine, C. Boucly, N. Magnin, P. Juteau, N. Voumard, CERN, Geneva, Switzerland

Abstract

Two LHC Beam Dumping Systems (LBDS) remove the counter-rotating beams safely from the collider during setting up of the accelerator, at the end of a physics run and in case of emergencies. Dump requests can come from 3 different sources: the machine protection system in emergency cases, the machine timing system for scheduled dumps or the LBDS itself in case of internal failures. These dump requests are synchronized with the 3 µs beam abort gap in a fail-safe redundant Trigger Synchronization Unit (TSU) based on a Digital Phase Locked Loop (DPLL), locked onto the LHC beam revolution frequency with a maximum phase error of 40 ns. The synchronized trigger pulses coming out of the TSU are then distributed to the high voltage generators of the beam dump kickers through a redundant fault-tolerant trigger distribution system. This paper describes the operational experience gained with the TSU since its commissioning with beam in 2009, and highlights the improvements, which have been implemented for a safer operation. This includes an increase of the diagnosis and monitoring functionalities and a more automated validation of the hardware and embedded firmware before deploying or executing a post-operational analysis of the TSU performance, after each dump action. In the light of this first experience the outcome of the external review performed in 2010 is presented. The lessons learnt on the project life cycle for the design of mission critical electronic modules are discussed.

INTRODUCTION

The TSU re-phases the dump requests with the circulating beam such that the 100 % rising edge of the magnetic field in the extraction kicker coincides with the end of the beam abort gap in the circulating beam [1].

Locked to the LHC beam revolution frequency, two redundant TSUs produce continuously dump trigger pulse trains synchronised with the beam abort gap. The distribution of these pulse trains is inhibited until a beam dump is requested. The first pulse which pass the inhibit stage are then sent via two redundant trigger fan-outs (TFO) to all the power trigger modules. In this way the first pulse after the reception of a dump request will synchronously trigger the system.

In case the beam revolution frequency is lost during more than one turn, an internally synchronised direct digital synthesiser based on a Numerically Controlled Oscillator (NCO) and on a Digital Phase Lock Loop (DPLL), precisely locked on the beam revolution frequency, will issue an internal dump request and will generate a dump trigger which is synchronous with the beam abort gap.

In addition, the two redundant TSUs continuously cross-check their DPLL synchronisation. A phase discrepancy greater than 40 ns between the two TSUs automatically issues a synchronous dump. If the synchronisation of only one of the TSU fails, a synchronous dump trigger is forced by the redundant system.

During the LHC cold check-out period all the functionalities of the TSU modules have been extensively tested and commissioned. No major technical issues have been identified at that stage. First operational experiences with beam have demonstrated the correct operational performance of the TSU module [2].

Nevertheless, some unforeseen events like asynchronous synchronous dumps* induced by the TSU modules have been recorded during the LHC commissioning. Despite a high level of embedded diagnosis and monitoring functionalities, tools to capture spurious synchronisation failures like synchronous asynchronous dump appear to be missing.

In addition, due to its key function within the critical LBDS system, it has also been decided to perform an external review of the TSU module and its embedded software in order to check and validate its functionality.

REVIEW

Early 2008, an internal review of the LBDS in general and of the trigger synchronisation and distribution system (TSDS) in particular has been conducted with the goal to validate all the implemented technical solutions before the start of the commissioning of the LBDS with beam. The final recommendation of this review has suggested to perform a more detailed review of the TSU module and its embedded software, and, to build a test bench for validation of its different functionalities.

Subsequently an external review of the TSU module has been performed in 2009/2010 by a company specialized in electronics engineering R&D, with the aim to obtain a detailed evaluation of the TSU module functional suitability and an identification of conception and/or implementation errors. The review objectives were:

- The validation of the correct implementation of the functional requirements,

* The TSU trigger outputs is asynchronous compared to the LHC revolution frequency but the 15 extraction magnets of the beam dump kicker are triggered in a synchronous way.

- The identification of possible hardware and software anomalies,
- The verification of the pre-series performance,
- The recommendation of possible improvements,
- The proposal of guidelines of possible maintenance procedures for the embedded software.

The review has been divided into three phases, requirement review, design review and hardware & software review, with an executive summary fully documented at the end of each phase.

The Requirement Review

The objective of the requirement review phase was to ascertain the adequacy of the requirements in defining the characteristics and the functionalities of the TSU module and to check that all requirements are covered by at least one module or sub-module of the TSU design, which could be in hardware, software or a mix of both.

A high level hierarchy that identifies all hardware and software modules with their corresponding functionalities has been created and cross-checked through a validation matrix with the list of all functional requirements.

The requirements review demonstrated that all requirements are indeed properly covered by at least one module and that all links between modules are coherent. However, it has been noted that the architecture of some required functions, like the PLL design, are sometimes too complex even although the hardware and software architectures are correct.

The Design Review

The second part of the review consisted in an in-depth analysis of the TSU module design in order to ensure that its conceptual design meets the baseline requirements and that its performance levels, typically reliability and availability, are within specification. A low level hierarchy has been created for this purpose and a requirement coverage matrix issued to synthesize the results with a basic OK/NOK status result for every requirement associated with a criticality level. A failure severity classification based on three levels of criticality has been used (Table 1)

Table 1: Criticality Levels

Critical	Failure that can induce a dysfunction considered as serious.
Major	Failure that can affect functionality considered as non-critical or with low probability.
Minor	Failure with no risk for the global functionality.

Over more than 200 functional requirements which were analysed the design review phase has identified 11 requirements as not properly implemented, from which only one has been classified as “critical”. Indeed it has been demonstrated that in case of an internal failure in one of the two redundant TSU modules, the faulty TSU indicates correctly its faulty state to the redundant module but doesn't generate any trigger, (in any case, an asynchronous dump should have been issued). Four other failure cases have been classified as major due to their possible impact on the module availability and the six remaining cases have been evaluated as minor

The Hardware and Software Review

An in-depth analysis of the TSU electronic circuit and its embedded software has concluded the review. The hardware has been completely checked with respect to modern state-of-the-art hardware design techniques as well as the embedded VHDL software. In addition, the major embedded software modules have been fully simulated and their reactions to incorrect operational conditions analysed in detail.

The analysis of the electronic circuit has highlighted the possibility of the propagation of an internal hardware failure to the redundant module, lack of protection against external environment perturbations, missing Electro-Static Discharge (ESD) and Electro-Magnetic Interference (EMI) protections and under-sized components in case of unforeseen DC mode operation. These design errors have been considered by CERN as minor taking into account the specific environment and operational conditions under which the TSU module is operated. Nevertheless, it has been decided to take these remarks into account for the development of the next hardware version of the TSU module.

The software analysis has highlighted missing debouncers to prevent meta-stable behaviour on some input signals and a wrong behaviour in case of a large phase jump of the beam revolution frequency signal that will imply an asynchronous synchronous dump output trigger. Both remarks have been confirmed by CERN as valid and implemented in a new release of the embedded software.

NEW TOOLS

On the basis of the first years of operation, two different tools have been developed in order to improve the maintenance, the monitoring and the post-operation diagnosis of the TSU.

An automated test bench has been developed to validate the hardware and software modifications before their operational deployment, and a trigger synchronisation unit internal post operational check (TSU IPOC) has been implemented with two functions: a monitoring function to acquire the sequence of synchronisation and triggering signals after each dump and a logic analyser of the dump events sequence for validation of its correct execution.

Automated Test Bench

When an update of the TSU module is required, the major difficulty during its re-commissioning comes from the need to have all the external conditions available in order to re-validate its correct functionality. As these conditions are only met when the machine is closed, a full

re-validation of the TSU, which is located in an underground gallery directly adjacent to the machine tunnel, is impossible.

Thus, An automated test bench based on a PXI crate has been developed to emulate all TSU external signals (dump request clients, beam revolution frequency, arming command…), to check the correct functionalities of the TSU through the application of different stimuli and error signals on the emulated signal and to validate the TSU response through the acquisition of the different TSU output signals, internal diagnosis registers and IPOC analysis result.

The test bench has been developed around a NI-PXI 8184 embedded controller running LabVIEW Real-Time. A NI-PXI 5412 arbitrary waveform generator has been used for emulation of the beam revolution frequency while the other signals are emulated through either an NI-PXI 5402 arbitrary function for frequency based signal or a NI-PXI 6115 multifunction I/O module for digital input.

All input functions and their actions are controlled through a LabVIEW based user interface (Figure 1). Through this interface, each individual signal can be independently tested or an automated test sequence can be launched.

When operated in automated test mode, the response to different failure cases on each input signal is tested. The test bench analyses all results and sets a flag on each test when passed. In the case of a failure, the test bench automatically records which rule has been violated.

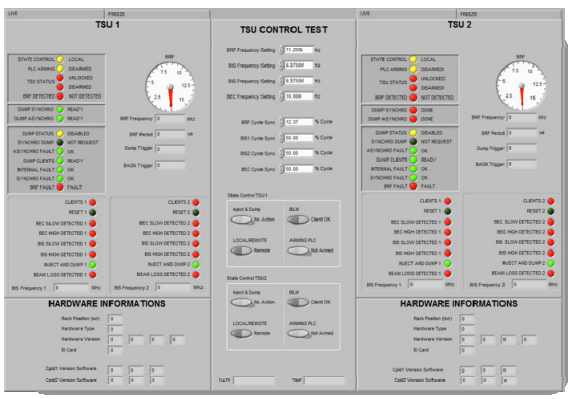


Figure 1: Test bench user interface.

TSU IPOC

The first release of the TSU embedded software includes detailed internal diagnosis functions. It appears rapidly that a tool to correlate in the time domain the response of the two redundant TSUs as well as the trigger signal distribution to the generators was missing.

A TSU IPOC with monitoring function has been developed to get a better understanding of the entire LBDS triggering process, from the capture of the dump requests up the monitoring of the synchronisation process and the generation and distribution of trigger and re-trigger signals.

The system is based on a PCI 32 bit 125 MS/s digital I/O module from SPECTRUM. The acquisition software is running on the LINUX front-end and has been implemented within the FESA framework.

Signals acquired after each dump include the dump requests, the beam revolution frequency, the re-phased beam revolution frequency, the synchronous trigger outputs from the TSU and the fan-out circuits as wel as the asynchronous trigger outputs and the re-trigger pulse chain signals.

A JAVA GUI permits an on-line analysis of the events sequence of the last dump and a replay previous dump for off-line analysis. An overview of a typical dump event sequence is shown in Figure 2.

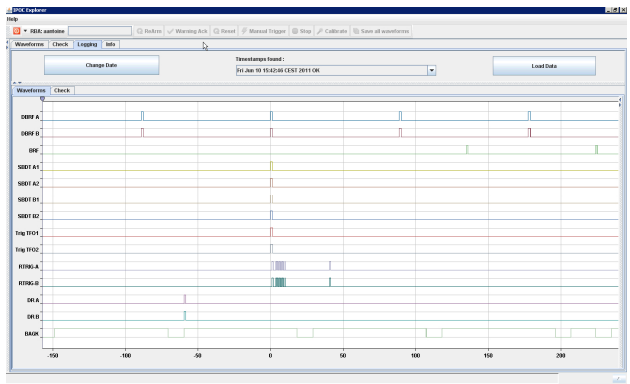


Figure 2: TSU IPOC Monitoring.

A second functionality of the TSU IPOC, based on signals acquired by the monitoring function, is the logic analyser function.. This automated analysis, performed after each dump, checks the correct sequence of triggering and re-triggering pulses, the correct delays between the different signals, the sanity of the signals, the correct frequency and phase offset between distributed and internally generated re-phased beam revolution frequency. For each analysis, a global and a detailed report is generated and archived. Figure 3 shows a typical dump event analysis summary.

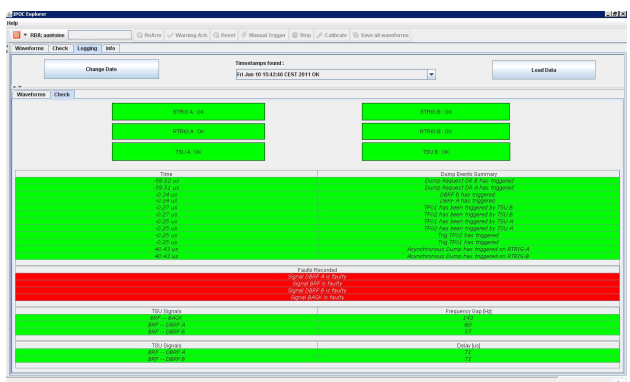


Figure 3: TSU IPOC logic analyser.

Up to now the IPOC has no link with the LBDS interlock system and its results don't affect the operational condition of the LBDS. In the future, this system will be included within the LBDS External Post Operational Check (XPOC) system [3] and thus force an expert to analyse and acknowledge failures before continuing with machine operation.

IMPROVEMENTS

Since its commissioning several minor updates on the TSU units have been applied successfully. Most of these updates have been induced by changes of specification, mainly to implement additional diagnosis. One has been made to remove a critical bug identified during the commissioning with beam. Here is a short summary of the different updates:

- Correction of a critical bug to avoid asynchronous synchronous trigger due to a hardware error in the design on the trigger output management output logic. This part, initially designed with combinational discrete components was responsible for a random phase shift of the synchronisation trigger pulse when the dump request was received;
- Implementation of the acquisition of UTC time stamp of the dump request;
- Implementation of a fast inhibit mechanism to LHC injection interlocking system in order to avoid the possibility to inject beam in the machine one turn after a beam dump;
- Consolidation of the TSU output signals for integration within the TSU logic analyser.

An important embedded software release has also been started for implementation of the external review recommendations. All critical and major identified failure cases were cured in this update, but the automated test failed. It was demonstrated that the change in the VHDL code induces a new routing of the FPGA circuitry, which revealed an insufficient filtering of the +1.2V power supply of the TSU board. A stronger inductance supported by a new decoupling capacitor was implemented and the following automated test bench passed with more than 35000 arming/dump sequences without any failure. That new release will be deployed at the end of 2011.

Finally, the review has taught us a new way of working for future projects. The project life cycle for a design, V cycle, consists on dividing the designers in two independent teams. The first team creates all the steps to design a final product (requirements, high level and low level architecture, etc.), while the second team is developing verification tools for all steps of the project to the final product, including the design of the final test bench. In that way, most of common modes in the conception are rejected which improves the reliability and the robustness of the final product.

CONCLUSION

The first operational experience and the external review led to the creation of three new functions. An automated test bench that allows the validation of any hardware or software modifications before deployment in operation, a TSU IPOC monitoring function with an on-line graphical interface to show a dump trigger signature and a TSU IPOC logic analyser looking at defined rules to validate the correct operation of the TSU units once in operation. Additionally, the external review has given us a new methodology in project design improving the reliability of final products, the V cycle.

The last release of the TSU units, taking into account all critical and major design errors highlighted by the external review, is now ready for deployment after a successful completion of the automated test process.

A new hardware design release will be started in 2012 to improve the robustness of the interfaces.

REFERENCES

[1] A. Antoine et al, "The LHC Beam Dumping System Trigger synchronisation and Distribution System", ICALEPCS'05, Geneva, October 2005.

[2] E. Carlier et al., "Commissioning of the Control System for the LHC Beam Dump Kicker System", ICALEPCS'09, Kobe, October 2009.

[3] N. Magnin et al., "External Post-Operational Checks for the LHC Beam Dumping System", ICALEPCS'11, Grenoble, October 2011.

LHC COLLIMATOR CONTROLS FOR A SAFE LHC OPERATION

S. Redaelli, R. Assmann, R. Losito, M. Donzé, A. Masi, CERN, Geneva, Switzerland

Abstract

The Large Hadron Collider (LHC) collimation system is designed to protect the machine against beam losses and consists of 108 collimators, 100 of which are movable, located along the 27 km long ring and in the transfer lines. The cleaning performance and machine protection role of the system depend critically on accurate jaw positioning. A fully redundant control system has been developed to ensure that the collimators dynamically follow optimum settings in all phases of the LHC operational cycle. Jaw positions and collimator gaps are interlocked against dump limits defined redundantly as functions of time, beam energy and the β^* functions, which describe the focusing property of the beams. In this paper, the architectural choices that guarantee a safe LHC operation are presented. Hardware and software implementations that ensure the required performance are described.

INTRODUCTION

The nominal beam stored energy at the Large Hadron Collider (LHC) will exceed 350 MJ, to be compared with the quench limit of super-conducting magnets of a few mJ per cm^3 and the damage limit of metal of a few hundred kJ per cm^3. The collimation system, based on 108 collimators located around the ring, protects the machine against beam losses and cleans the beam halos [1].

Each collimator has two jaws controlled by four stepping motors to precisely adjust jaw position and angle with respect to the beam. Stepping motors have been used to ensure high reproducibility of settings. Linear Variable Differential Transformer (LVDT) sensors and resolvers have been installed to monitor the position of the axes of the collimators in real-time (RT) at 100 Hz. The collimator jaws follow motion profiles expressed as functions of time, which is a unique feature for collimation systems in particle accelerators. Different sets of functions are optimized for the different LHC operational phases. Resolvers are used to detect losses of motor steps whereas LVDT readings are compared redundantly with safety limits. Limits a defined as functions of time, of beam energy and of β^* functions that express the focusing property of the beams. If the measured axis position violates any of these limits, which are always active in parallel, the low level control system requests an immediate abort of the circulating beams.

The control system of the collimators is responsible for the motion control, synchronization and survey of about 400 axes (see Tab. 1). It is characterized by challenging requirements [2] such as timing synchronization in the motion axes at the microsecond level; motion repeatability of a few micrometers and accuracy in the monitoring of the profile in execution. High reliability is ensured through architectural choices and redundancy implemented to ensure machine safety. In the next section the architecture of the LHC collimator control system is described. The redundant strategy implemented on the collimator axes monitoring and survey through interlock limit functions is then presented. An analysis of the additional interlock conditions is presented in the last section.

Table 1: Main System Parameters. The number of settings in the second part of the table is calculated for the 2011 operational cycle, with squeeze to $\beta^* = 1$m in IP1 and IP5. Energy– and β^*–limits are common to all machine cycles (same values resident in the hardware). The two dump protection collimator are not included in this list.

Parameters	Number
Movable collimators in the ring	85
Transfer line collimators	13
Stepping motors	392
Resolvers	392
Position/gap measurements	584
Interlocked position sensors	584
Motor settings versus time	1760
Threshold settings versus time	3054
Threshold settings versus energy	196
Threshold settings versus β^*	384

COLLIMATOR CONTROLS

The Control Architecture

In Fig. 1 the general layout of the LHC collimator control system (LCCS) is presented [3]. Starting from the bottom we can identify the following layers: i) Low level front-ends based on two National Instruments PXI systems for the motion control and survey ii) the collimator middle ware based on a gateway that concentrates all the data accesses from the top level application via a standard CERN middle ware server [4] and establishes peer to peer connections with the collimators' low level control systems through the Data Interchange Management protocol (DIM) [5]; iii) the Central Control Application (CCA) [6] is responsible for generating and orchestrating the settings for the whole system and for sending them to the middle ware referring to the collimators' FESA class [7]. The CCA is fully integrated into the LHC Software Architecture (LSA) environment [8].

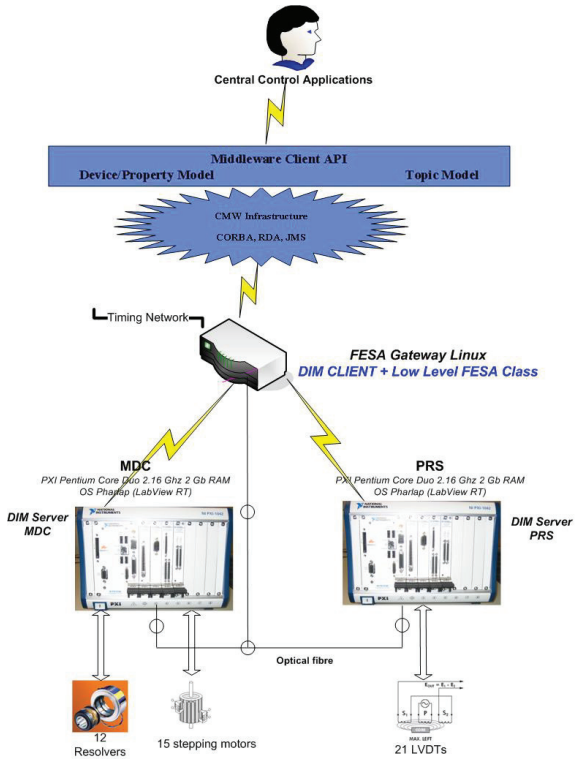


Figure 1: Architecture of the LHC collimator control system.

Low-level Control

Controls choice PXI platforms from National Instruments running LabView RT have been chosen as low level RT control system. Significant improvements of the PXI systems were applied to enhance their reliability and robustness [9]. The main changes are: i) Diskless controller equipped with the PXE network boot protocol; ii) Double core CPU to split the computational load; iii) Implementation of watch dog timers on the CPU and FPGA to detect stuck conditions on the host and bad working of the FPGA; iv) Monitoring of the system parameters (i.e. memory usage, CPU and chassis temperature, CPU load) to prevent operational anomalies (i.e. memory leakages). The control system's reliability was increased by splitting the functions of motion control and survey on two independent PXI systems: the Motor Drive Control (MDC), responsible for the generation of stepping pulses and for the resolver monitoring for up to three collimators, and the Position Readout and Survey (PRS), responsible for the synchronous monitoring of up to three collimators via the LVDTs.

Motor driver controller (MDC) The MDC receives motion commands (i.e. simple displacements or long motion profiles) from the top level through the FESA middle ware. It verifies the consistency of the requested settings against present status of the collimator, then checks for steps lost during execution in RT using one FPGA card per collimator. A specific software was developed for the step generation: in the host controller, the motion profiles are interpolated with a 5 μm step resolution to generate the set points; these are sent via a FIFO to the FPGA, where a step generation loop, operating at 1 MHz, produces the pulses for each collimator motor. Each motor's resolver is read synchronously with the generated steps at up to 400 Hz thanks to a custom reading solution based on CORDIC transformations [3]. The nominal speed for discrete movements is 400 steps/s, which corresponds to 2 mm/s.

Position readout system (PRS) The PRS is responsible to verify that the collimator jaw positions and gaps are within the safety thresholds and to trigger a beam abort otherwise. On each collimator, 6 LVDT sensors are installed to read the 4 axis position and the 2 upstream and downstream gaps. The LVDT position sensors for up to 3 collimators are read at a frequency of 100 Hz and with an accuracy of a few μm [2]. Two parallel 16 bit ADC cards sample the secondary voltages of the 7 LVDTs of each collimator. A sine fit algorithm, which is properly optimized for RT implementation, runs on the Host and estimates the amplitudes. A ratiometric technique is then used to obtain the position. The survey process also runs on the Host but the synchronization is ensured by timing signals generated on an FPGA card and passed via the PXI bus [3].

Collimator Middle-ware

A gateway is installed in each LHC point with collimators to supervise and synchronize all the systems of that point. The RT actions (e.g. MDC motion or PRS monitoring start) are triggered through pulses sent via optical fibers directly to the PXI FPGA cards. All the gateways are equipped with a CERN timing receiver and synchronized via the CERN timing network [10]. This provides not only the LHC timestamps, but also machine status information (i.e. beam energy, β^*) [11]. The information of energy and β^* are used to determine via special tables the gap limits to be sent to the PRS over the network at 1 Hz refresh rate. On the PRS specific watch dog timers have been implemented to detect network communication problems and use, in this case, the tightest limits. On each gateway a FESA server exposes each collimator's data to the operator applications via a device model with information organized in properties and data fields. On the other side, a DIM client running on the same gateway fetches data and sends commands to the DIM servers on the corresponding PXI (MDC and PRS).

POSITION LIMITS AND INTERLOCKS

Operation Cycle for Collimators

The sequence of the relevant phases through which the LHC is driven to establish collisions for physics data taking is referred to as the operational cycle. This includes the injection of high intensity beams, the energy ramp, the betatron squeeze – when the focusing properties of the lattice are changed to reduce the β functions at the collision

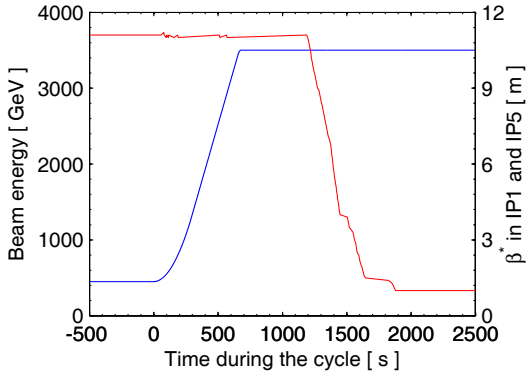


Figure 2: Beam energy (blue, left axis) and β^* in IP1 and IP5 (red, right axis) as a function of time during the LHC cycle. Time zero represent the start of the energy ramp.

points – the establishment of collisions, the physics data taking and the pre-cycle. Due to the large LHC stored energy (more than 20 MJ already at injection), beam collimation is needed in all phases. Appropriate redundancy of interlocks has been built into the system in order to ensure that the critical phases like the energy ramp and squeeze are performed with collimators at the correct positions. For this purpose, the concepts of energy– and β^*–limit functions have been built into the system in addition to standard limits as a function of time.

The beam energy and the β^* functions in a high-luminosity experiment are shown as a function of time during an LHC cycle in Fig. 2. Not shown in Fig. 2 is the pre-cycle without beam when the injection conditions are restored after a beam dump at high energy. Correspondingly, the collimators are moved as shown in Fig. 3. As representative examples, one primary collimator (TCP) of the betatron cleaning insertion and a tertiary collimator (TCT) that protects ATLAS, are shown. The measured collimator gaps and the different types of interlock limits are given as a function of time. Only dump limits, and not the warning limits, are shown.

Limit Functions Versus Time

Most accelerator systems are driven with functions of a pre-defined time duration only during energy ramp, betatron squeeze and collision preparation. The function duration is determined by the property of the power converters. A specific feature of the LHC collimators is that their jaws can be moved with pre-defined functions of time. This feature is necessary to allow the optimum settings to be maintained throughout the operational cycle, when the beam energy or the machine's optics change [12]. Correspondingly, limit functions versus time are also defined for each motor axis and for a gap. Inner and outer limits, with additional operational warning levels are defined for each degree of freedom, for a total of 24 limits functions for collimator, with a clear redundancy, since the degrees of freedom are only 4, one for each motor of the collimator.

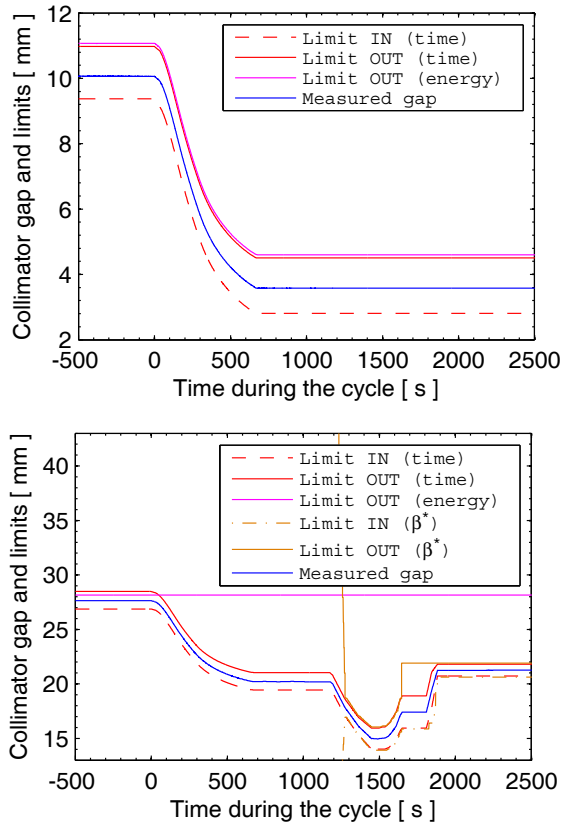


Figure 3: Gap and interlock limits versus time for a primary collimator of the betatron cleaning insertion (top) and for a tertiary collimator in IP1 (bottom) during a complete operational cycle (see Fig. 2).

Discrete Limits

Outside cycle phases driven by functions, the collimators remain idle at constant settings and discrete changes ("trims") are possible. In these cases, discrete collimator limits are used. These limits are computed from the functions: for example, the injection settings correspond to the ramp function's first point, and the limits during physics are given by the collision function's last points. Discrete limits apply for the motor axes and gaps, for a total of 12 dump limits and 12 warning limits per collimator.

Energy Limits

The limits as a function of energy were conceived to ensure that the collimator gaps follow the reduction of beam size during the energy ramp. Maximum allowed gap values versus energy are defined for each gap LVDT. The same concept is used for injection protection collimators – even if the are not "ramped" – to ensure that injection is not possible is collimator gaps are larger than safe limits [13].

β^* Limits

Additional inner and outer limits as a function of β^* are checked for upstream and downstream gap measurements (4 limits per collimator) in order to make sure that the ter-

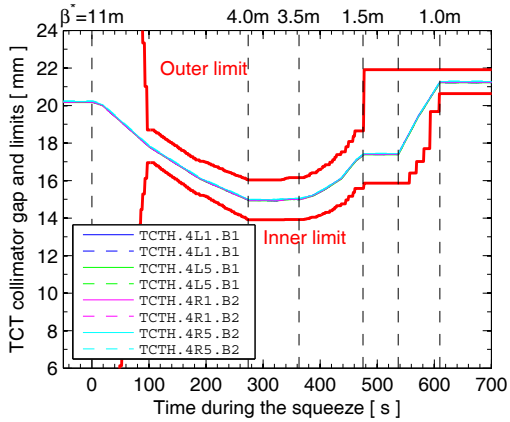


Figure 4: Measured gap position and β^*–limits of 2 tertiary collimators in IP1 and IP5 versus time during a betatron squeeze down to 1 m.

tiary collimators in the experimental region move as required while the optics is changed. The β^* information is distributed for each interaction point (IP) separately. Each collimator can be configured to use as input the β^* in any IP or the minimum of the 4 IPs. An example for 2 TCTs in IP1 and IP5 is given in Fig. 4.

Strategy for Motor Blockage

The controllers of each motor axis are blocked upon reaching the discrete or functional limits versus time in order to avoid that a collimator jaw runs into the circulating beam in the extremely unlikely case that the beam abort was not triggered by the violation of the inner position limits. This blockage mechanism also reduces the risks of mechanical damage in case of erroneous manipulations, like setting of tilt angles above the mechanical limit of 2 mrad or requested position beyond the mechanical end stops that could be reached in case of end switch failures.

Unlike the time-dependent limits, the limits as a function of energy and β^* do not block the motors. This allows the jaws to reach the open parking positions during the recycle without beam. In this phase, the energy limits generate an interlock that prevents injection of unsafe beams until the collimators are moved to safe injection settings. At every operational cycle, it is also verified that all the connections between the collimators and the beam interlock system are operational.

COLLIMATOR STATUS INTERLOCKS

In addition to the position interlocks, the PRS unit can also dump the beams in a number of cases when the machine protection role of the system cannot be ensured:

- Reboots of the low-level systems;
- Power cuts that affect the PRS PXI;
- Set of "Local" mode that allows expert checks and sensor calibrations;
- Stuck conditions detected on the PRS CPU through watch dog timer verified on the PRS FPGA.

CONCLUSIONS

In this paper the LHC Collimator control system has been presented focussing on design, architectural choices and control strategies that guarantee a safe LHC operation. In all operational cases, a highly redundant survey of collimator position ensures that the system is at the required safe settings. This strategy has been successfully validated by 2 years of LHC operation with high intensity beams.

ACKNOWLEDGEMENTS

The authors would like to thank M. Jonker, M. Sobczak for the discussions on the collimation control architecture. They are also indebted to a number of people in National Instruments, in particular B. Runnels, C. Loew, D. Shepard, G. Cirigioni, S. Concezzi, C. Moser, C. Farmer. The LHC operation (in particular M. Lamont and J. Wenninger), the machine protection, the controls (G. Kruk) and the timing teams are also kindly acknowledged.

REFERENCES

[1] R. Assmann, "Collimation for the LHC High Intensity Beams", Proceedings of HB2010.

[2] A. Masi *et al.*, "Measured performance of the LHC Collimators low level control system", Icaleps (2009).

[3] A. Masi and R. Losito, "LHC Collimator Lower Level Control System," 15th IEEE NPSS Real Time Conf. 2007.

[4] M. Arruat *et al.*, "CERN front-end software architecture for accelerator controls", (CERN-AB-2003-101-CO), 4 p. (2003).

[5] C. Gaspar, and M. Donszelmann,"DIM a distributed information management system for the delphi experiment at CERN", IEEE 8th Conf. Real Time Comput. Appl. Nucl., Particle Plasma Phys., 156158. Vancouver, Canada. (1993).

[6] S. Redaelli *et al.*, CERN EDMS document LHC-TCT-ES-0001-10-00 (2007).

[7] S. Redaelli and A. Masi, "Middle-level interface to control movable devices like the LHC collimators" LHC-TC-ES-0002-20-00 (2008).

[8] G. Kruk *et al.*, "LHC software architecture (LSA) – evolution towards LHC beam commissioning", ICALEPCS, 2007.

[9] A. Masi *et al.*, "Reliability review of the LHC collimators low level control system", IFAC Symp. on Large scale systems (2010).

[10] P. Alvarez *et al.*, "Nanosecond level UTC timing generation and stamping in CERN's LHC", (CERN-AB-2003-111-CO), 4 p.(2003).

[11] B. Todd, "Safe machine parameters 3V0", LHC-CI-ES-0004-01-10 (2010).

[12] R. Bruce *et al.*, "Principles for generation of time-dependent collimator settings during the LHC cycle", IPAC,(2011)

[13] S. Redaelli *et al.*, "2011 modifications of the LHC collimator controls relevant for machine protection", LHC-OP-MPS-0016, EDMS doc. 1119832 (2011).

QUALITY-SAFETY MANAGEMENT AND PROTECTIVE SYSTEMS FOR SPES

D. Benini, S. Canella, INFN-LNL, Legnaro, Italy

Abstract

The realization of a nuclear facility for the production of radioactive ion beams requires a deep study of the safety aspect: an high degree of reliability in the Protective System must be achieved to prevent hazardous situations for operators, population and the surrounding environment.

For the INFN SPES project, a *Quality and Safety Management System* is going to be realized. In this work we will present its general structure, functions and goals. We will then focus our attention on the Access Control and Dose monitoring systems which are the key features of the SPES *Protective System* in the framework of the QSMS.

INTRODUCTION

SPES (*Selective Production of Exotic Species*) is a INFN project for the realization of a Radioactive Ion Beam facility at Legnaro National Laboratory (LNL).

The radioactive beams production method is based on the ISOL technique (*Isotope Separation On Line*). The exotic isotopes are extracted from proton induced fission products in a UCx direct target, typical expected fission rates are of 10^{13} fission/s. The SPES proton driver is a two exit port cyclotron with a variable energy from 15 MeV up to 70 MeV, the foreseen maximum current value is 0,750 mA. A general layout of the SPES facility is illustrated in Fig. 1.

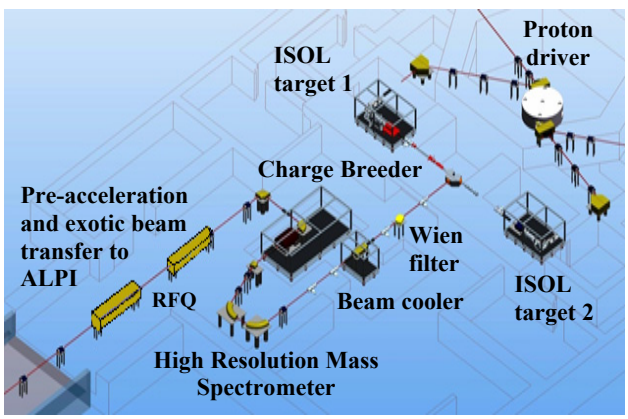


Figure 1: SPES layout.

An overall tool for managing the safety issues of SPES is the Quality and Safety Management System (QSMS), its development is one of our goals in the next future. The implementation of this System starts from the definition of the safety and quality policies to be achieved and involves the drafting of a hierarchical framework of procedures. The Access Control and the Dose Monitoring Systems will be also analyzed for the production of specific documentation.

The Protective System of SPES will be designed with the necessary intent of achieving a high level of safety and reliability, in this way dangerous situations for people and the environment will be avoided. In this paper we present the QSMS concept and its software implementation, the specific features of the SPES Protective System will also be described.

THE QSMS OF SPES

The QSMS is a managing tool that LNL has chosen to realize for handling all the phases of SPES starting from the initial stage of the facility design.

It is implemented according to Italian laws, technical standards and all mandatory prescriptions that the project must comply. Moreover the international standard ISO 9001:2008 for Quality and OHSAS 18001:2007 for Safety will be the main reference for the development of the QSMS.

An Environmental Management System (ISO 14001:2004 compliant) is already operative at LNL: the QSMS of SPES will be developed considering its structure and also the possible future integration between the two systems.

The first step for the implementation of the System is the definition of a policy of Quality and Safety for SPES, it will be pursued through the definition and the realization of specific objectives. The approach of the QSMS is to divide the SPES project in the following 5 phases of its lifecycle:

- Design;
- Realization;
- Operation;
- Maintenance;
- Disposal.

Every stage will be analysed to identify the activities that should be controlled to guarantee safety and quality for the SPES facility: starting from general guidelines the specific operating instructions for each area will be identified. Some of the aspects that will be analysed for the phases of design and construction concern, for example, the collection of all technical drawings, the identification of legal and technical requirements and the description of the techniques for hazards and risks analysis. Afterwards all the ordinary and emergency procedures for the last three phases (operation, maintenance and disposal) will be drawn.

The complete documentation concerning the QSMS is collected and catalogued according to specific storage rules that allow a fast identification of each document and an easy retrieval in the electronic archive. Documents are hierarchically organized according to a pyramidal scheme: at the top we find the general paper describing the QSMS in its whole, moving downward there are more

precise documents explaining with more details each specific activity.

Figure 2 shows the documents organization scheme of QSMS.

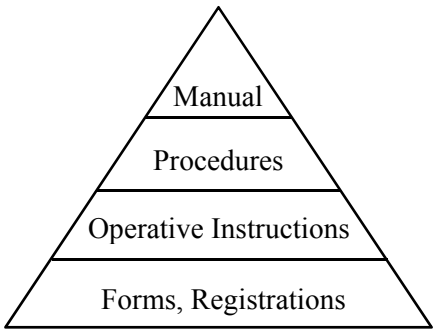


Figure 2: Scheme of the QSMS documentation.

Documents are divided in the following categories:

- Manual: is the text describing the QSMS general features explaining its purposes and organization modalities;
- Procedures: these documents describe how every single activity foreseen by the System has to be performed. They are divided in Managerial and Technical considering their connection to the organization of the management system rather than specific SPES activities;
- Operative instructions: they refer to specific procedures and contain a detailed description of how a specific activity has to be performed;
- Forms and registrations: these are report documents to be filled at the end of each activity. They are intended for operations logging.

Procedures for the Risk Analysis and the SPES Protective System

In this work we want to focus the attention on the procedures of the QSMS related to the Risk Analysis of the plant and the Protective System of SPES.

Concerning the Risk Analysis a detailed identification of the hazards and a consequent evaluation of all the risks is needed to optimize both plant design and safety systems. From the point of view of the necessary documents we are preparing a procedure describing the standard techniques that must be applied, the plant and the specific components that must be analysed, the data reports standards. The investigation method requires the use of the following techniques:

- Failure Mode and Effect Analysis (FMEA): with this method one looks for any possible failure of the plant of interest, finding out its gravity and frequency of occurrence. This is done to underline possible design errors of the plant or the presence of single components loosely reliable.
- HAZard and OPerability analysis (HAZOP): process variables are examined looking at the deviations from standard operative conditions. This is done to evidence possible failing situations of the system which can lead to an accident (called Top Event, TE).
- Fault Tree analysis (FT): it evaluates the frequency of occurrence of the TEs evidenced with the HAZOP method.

The results obtained from the combined use of this three methods give an important input for the design and the realization of the SPES Protective System. The guidelines indicated by the LNL's radiation protection officer will be another reference point to take into account, this kind of requirements will have to be completely fulfilled.

From the point of view of the QSMS the documents for the SPES Protective System will contain a detailed description of the safety disposals organization and the single procedures (ordinary and of emergency) to be followed during the facility operation.

THE SOFTWARE OF THE QSMS

The management of the data flow foreseen by the SPES QSMS will be realized through a custom software. This tool is designed to facilitate the application of the procedures of the QSMS during the different phases of the facility lifecycle. It will have to act as data collector during the project realization as well as an automation tool for the working and maintenance procedures.

We intend to realize a user-friendly interface enabling every-day use, easy procedures retrieval and fast actions logging. Flexibility is also one of the most important features, for this reason a modular structure will be implemented so that each unit can be developed, created or cancelled as needed.

Technical Description

From the technical point of view, the software is composed of two parts:

- a Relational Database Management System (RDBMS) for data collection;
- an interface that allows users interaction with the data stored in the database. To allow easy distribution of the software and multi-platform support, a web-based application has to be preferred.

To realize this project a dedicated website will be realized on proper LNL servers using the standard Apache/PHP/MySQL open source platforms.

The database is going to be divided in the following main units:

- SPES elements* of these four main areas: experimental plant, laboratories, safety systems, infrastructures (buildings, plant);
- Personnel;
- Documents;
- Stocks;
- Suppliers and other contacts;
- Laws and technical standards;

* Elements are all the equipment, plant and infrastructures in which the project SPES can be subdivided.

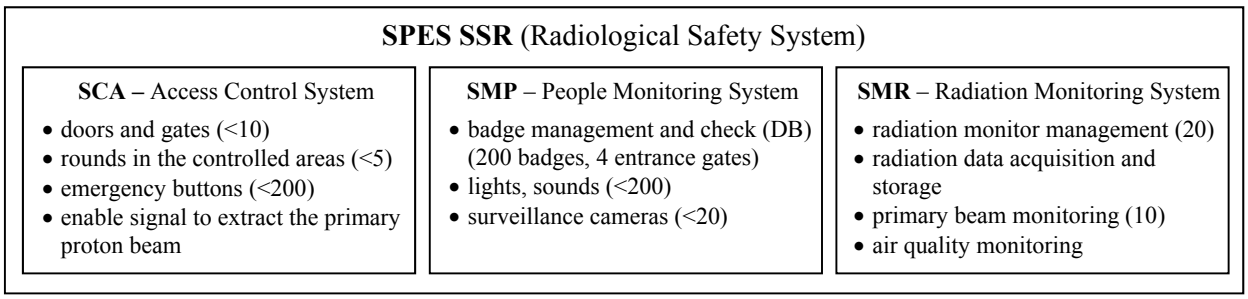


Figure 3: The three elements of SSR with the foreseen number of objects to be managed.

- Procedures (operation of equipment, maintenance, emergencies, accidents, etc.);
- Audit.

In the framework of the QSMS we will develop the SPES Protective System illustrated in the following paragraph.

THE SPES PROTECTIVE SYSTEM

The SPES Protective System will be an instrumented system with the function to detect potentially dangerous conditions for people and the environment, when such a state is detected it will have to execute a sequence of actions to restore a safe state.

For this goal, the SPES protective system will be characterized by the following attributes:

- independence from the process managing the plant, that is from the SPES control system;
- its integrity and functionality has to be maintainable and always be verifiable;
- high reliability (redundancy, at least in the most critical parts for safety);
- any access to it (for checks or maintenance) is to be performed under surveillance and monitored;
- changes are always to be logged, approved and promptly documented.

These attributes have to be maintained by good administration practices (i.e. by QSMS).

The Structure of the Protective System

The structure of SPES Protective System will rely on two interconnected sub-systems: one for conventional, the other for radiological risks. During the operating periods, a Conventional Safety System (**SSC** - Sistema di Sicurezza Convenzionale) will avoid/reduce the risks related to standard engineering (high voltage, water cooling, venting systems). The Radiological Safety System (**SSR** – Sistema di Sicurezza Radiologica) will avoid/reduce the following risks:

- irradiation and contamination of people both when SPES is running and when it is off (activated materials may still be present in the plant site);
- any case of uncontrolled sprinkling of radioactive material outside the SPES complex.

Both the SSC and SSR must avoid/reduce the safety risks also when SPES control systems and the protective systems themselves are off or in fault: they must be fail-safe.

With more details, SSR will be made up by three elements:

- an Access Control System (Sistema Controllo Accessi – **SCA**) for gates and doors, for rounds in the controlled areas, for emergency buttons and for the final enabling control signal to the primary proton driver;
- a People Monitoring System (Sistema di Monitoraggio delle Persone – **SMP**), to enable/disable people access to different building areas (by badges/tokens), to manage warning lights and tables, sounds, surveillance cameras;
- a Radiation Monitoring System (Sistema di Monitoraggio delle Radiazioni – **SMR**), to control radiation monitors, to acquire and store related data, to manage thresholds and produce alarms and interlocks, to monitor the primary proton beam quality, to check the activation level of pumped air to be eventually released outside the SPES complex.

Figure 3 shows a scheme of the SSR with its elements.

The whole SCA system will be redundant at least at level 2: two parallel systems will acquire the same signals and produce the final enable/disable output for the primary beam accelerator. In particular, a first system will be based on safety-oriented PLC architecture, while a second one, minimal but highly reliable, will be made of embedded ("custom") not-programmable logic cards (for example with FPGA based logic modules).

The SCA system will have to produce the enabling signal for the proton beam extraction from the cyclotron. Its high reliability will be based on redundancy and high quality of the hardware system for the IN/OUT signals from the field (gate and door status, round and emergency buttons) and to the actuators which will be produced by redundant detectors and command chains (at least 2 for each gate and door and for the main beam shutters), separately cabled.

For the SMP section and SMR (radiation monitors), a highly reliable PLC based architecture should be sufficient.

Any fault or anomalous behavior in any of these sub-systems will have as direct consequence the switching off of the primary beam.

EXTERNAL POST-OPERATIONAL CHECKS FOR THE LHC BEAM DUMPING SYSTEM

N. Magnin, V. Baggiolini, E. Carlier, B. Goddard, R. Gorbonosov,
D. Khasbulatov, J. Uythoven, M. Zerlauth, CERN, Geneva, Switzerland

Abstract

The LHC Beam Dumping System (LBDS) is a critical part of the LHC machine protection system. After every LHC beam dump action the various signals and transient data recordings of the beam dumping control systems and beam instrumentation measurements are automatically analysed by the eXternal Post-Operational Checks (XPOC) system to verify the correct execution of the dump action and the integrity of the related equipment. This software system complements the LHC machine protection hardware, and has to ascertain that the beam dumping system is 'as good as new' before the start of the next operational cycle. This is the only way by which the stringent reliability requirements can be met.

The XPOC system has been developed within the framework of the LHC "Post-Mortem" system, allowing highly dependable data acquisition, data archiving, live analysis of acquired data and replay of previously recorded events. It is composed of various analysis modules, each one dedicated to the analysis of measurements coming from specific equipment.

This paper describes the global architecture of the XPOC system and gives examples of the analyses performed by some of the most important analysis modules. It explains the integration of the XPOC into the LHC control infrastructure along with its integration into the decision chain to allow proceeding with beam operation. Finally, it discusses the operational experience with the XPOC system acquired during the first years of LHC operation, and illustrates examples of internal system faults or abnormal beam dump executions which it has detected.

INTRODUCTION

The LHC Beam Dumping System (LBDS) must insure the loss-free extraction of the two circulating beams over one LHC revolution, at a programmed dump at the end of a physics run, or in case of emergency.

LBDS Kicker System

The LBDS kicker system consists of 15 horizontal extraction kickers (MKD), 15 vertically deflecting septum magnets (MSD) to deflect the beam into the extraction channel, and 4 horizontal and 6 vertical dilution kickers (MKB) to dilute the beam before it reaches the absorber block (TDE) situated around 1 km away from the extraction point in the LHC ring [1].

LBDS Kicker Control System

During LHC operation, the kicker generator voltages must follow the beam energy, which can vary from 450 GeV at injection up to 7 TeV at collision energy. Moreover, the LBDS triggers must always be issued synchronously with the 3 μs beam-free abort gap that allows the magnetic field of MKDs to reach their nominal strength. To guarantee faultless operation, various fail-safe or fault-tolerant (redundant) control sub-systems have been put in place (SCSS, BETS, TSDS, FAAS) [2]. Each of these sub-systems has an Internal Post-Operation Check (IPOC) system which verifies the correct operation during the latest beam dump action. We can mention the following IPOC systems:

- The IPOC FAAS [3] systems acquire the MKD and MKB current waveforms, analyse them, and save them into the Post Mortem Data Collection storage [4].
- The Trigger Synchronisation Unit (TSU) IPOC systems [5] acquire the waveforms of the main LBDS synchronisation signals as well as all redundant trigger signals. These waveforms are analysed to check the correct synchronisation of all signals, and to ascertain that all redundant signal paths were operational. The TSU IPOC systems will save their analysis results into the Post Mortem Data Collection storage.

Beam Instrumentation

Various beam instruments are deployed at LBDS for diagnosis purpose [1]:

- Beam Loss Monitors (BLM) in the dump line and the extraction area;
- Beam Current Transformers (BCT) on the LHC ring and in the injection and extraction channels;
- Beam screen (BTV) in front of the TDE;
- Beam Position Monitors (BPM) in the extraction channels;
- Beam Synchrotron Radiation monitor for beam abort gap population (BSRA).

All beam instrumentation equipment provide their post operation data, consisting of history buffers of states and measurements at the moment of the beam dump action.

XPOC System

After each dump action, the sanity of the LBDS is verified by the XPOC system through an in-depth analysis of the LBDS performance. The XPOC analysis server will process all the post operation data stored by the various LBDS control sub-systems or beam instrumentation equipment, to make sure that all LBDS equipment performed correctly, and that the beam was extracted as expected, in a manner to ascertain that the

beam dumping system is 'as good as new' for the next operational cycle.

XPOC SYSTEM IMPLEMENTATION

The XPOC system has been programmed with Java 6, using the standard technologies selected for the CERN control application development like Swing, the Spring framework and the Controls Middleware (CMW) [6]. It mainly relies on the LHC Post Mortem Analysis Framework (PMA) [4] which provides:

- A powerful and reliable data collection system to store PM data buffers coming from all equipment;
- An event builder system to group collected PM data buffers into PM events;
- An analysis system which launches the various analysis modules for each PM event;
- Connections to other LHC control systems such as the Logging Database, the Software Interlock System and the LHC Sequencer;
- A Graphical User Interface application to display analysis results of the latest PM events, or to replay any past PM event.

The XPOC system has been deployed in the form of three parts: XPOC data collection server, XPOC analysis server, and XPOC-Viewer user interface application used for the monitoring and control of XPOC analysis server.

The two XPOC server processes run on the same computer HP ProLiant BL460c G7 equipped with two Intel Xeon(R) CPU X560@2.8GHz (with a total 24 of cores), 24 GB RAM, and running Scientific Linux CERN 5 (SLC5) operating system.

XPOC Data Collection Server

Several of the beam instrumentation devices do not use the standard PM data collection mechanism which would allow them to directly send their PM data buffers to the PM data collection server. Therefore, a special XPOC data collection server had to be developed, which uses the normal CMW subscription/notification mechanism to obtain the necessary data.

After each 'beam dumped' event received from the timing system, the XPOC data collection server starts to collect PM data buffers from a predefined list of equipment and sends them to the PM data collection server.

When all PM data buffers for a 'beam dumped' event have been collected, the XPOC data collection server builds an XPOC PM event with the predefined list of collected PM data buffers attached. Two types of PM events are generated: 'XPOC-Beam1' and 'XPOC-Beam2'; dumps of LHC beam1 and beam2 will be handled separately.

XPOC Analysis Server

Once an XPOC PM event has been built, the XPOC analysis server starts a new analysis session and launches the execution of the equipment dedicated analysis modules in parallel threads, following their configured dependencies.

After execution of all analysis modules, the XPOC analysis server publishes all module result data to the LHC Logging Database.

The overall status of the XPOC analysis session is sent to the LHC Sequencer and Software Interlock System.

User Interface

The XPOC-Viewer application shows the latest 'XPOC-Beam1' and 'XPOC-Beam2' PM event analysis results, and allows the reset of SIS latches by authorized people.

A screenshot of XPOC-Viewer is shown in Figure 1

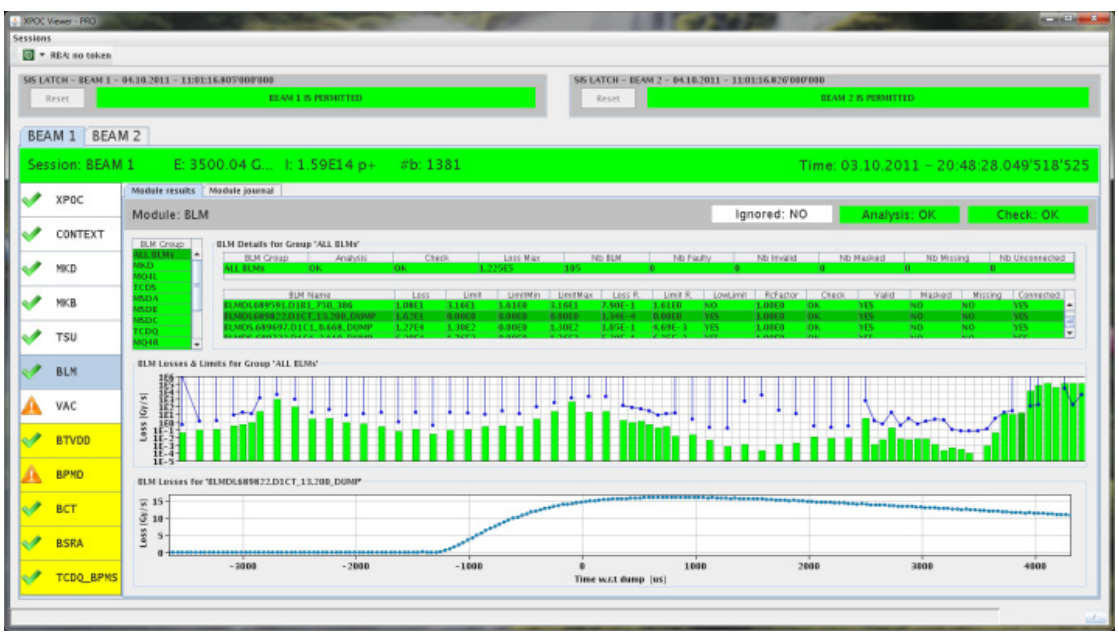


Figure 1: XPOC-Viewer application displaying status of XPOC SIS latches and BLM analysis module results.

XPOC ANALYSIS MODULES

Modules Dependencies

The list of all XPOC analysis modules is visible in the left column of Figure 1. The "CONTEXT" analysis module, which is a dependency for all other analysis modules, is executed first. Then the execution of all other analysis modules is launched in parallel, and finally the "XPOC" analysis module is executed to perform the overall analysis and give the final XPOC analysis session result.

CONTEXT Module

The aim of this module is to compute the "dumped beam intensity" and the "dumped beam filling pattern" present in the machine at the time of dump, as well as to determine the precise "dump timestamp". This information will be used by the other XPOC analysis modules to validate their input PM data buffers based on their timestamp, and to determine limits and thresholds to check data analysis results against.

The timestamp is taken from the LBDS trigger timestamp with a precision of about 100 ns (1 ns resolution). If no TSU data are available, the timestamp is taken from the PM event, which is less accurate (1 ms resolution).

The beam intensity calculation is based on three pieces of information:

- The history buffer of 30 s of the circulating beam intensity, with a sampling rate of 20 ms;
- The timestamp of the latest injection in the LHC;
- The beam intensity of the latest injection in LHC.

The circulating beam intensity is determined by performing an average over a few samples from the history buffer just before the dump action. Then the timestamp of the latest injection in LHC is compared to the dump timestamp to check if the dump took place just after injection. If a difference less than 20 ms is detected, the beam intensity of the latest injection in LHC is added to the circulating beam intensity, and the injected bunch pattern is added to the circulating beam pattern.

TSU Module

This module makes sure that the TSU cards and the trigger distribution systems functioned as expected, by checking that:

- Both TSU cards have properly detected the dump request and issued a synchronous dump trigger in less than one LHC revolution;
- The TSU IPOC has successfully checked the presence and the correct synchronisation of all LBDS synchronisation and redundant trigger signals.

MKD and MKB Modules

These two modules analyse the current waveforms in the dump kicker magnets (MKDs) and dilution kicker magnets (MKBs), recorded by the six IPOC FAAS [3]. The analysis consists of finding the critical points on the waveform to determine its characteristic values, like delay, rise time, strength, and check that these values are within predefined limits determined during the LBDS kicker calibration measurements [3].

BLM Module

This module performs the analysis of losses recorded by some BLM installed near LBDS equipment and near collimators all around the LHC. It uses the loss history buffer (+/- 3ms around the dump event) provided by every BLM device to determine their maximum loss at the moment of the dump action. It then computes loss limits based on beam energy and beam intensity provided by the "Context" analysis module. As there are quite a lot of BLM devices, they are grouped by family to easily manage their limits and write out to logging the maximum loss value per family. Each family has a reference BLM with defined ratios of loss versus energy and loss versus intensity, which allows the calculation of loss limit for a given dump action. All other BLMs in the same family have a constant loss limit ratio with respect to their family reference BLM. These limit ratios are determined by analysing the logged BLM XPOC analysis results for many 'normal' dumps at various energies and intensities. The measured losses are compared to the limits for each BLM to make sure that they all are within the computed limits.

Other Modules

Some modules are still under evaluation, so they are ignored by the final XPOC check module (their names are displayed in yellow in Figure 1). They are normally executed, and their results are displayed and logged as they provide very useful information for the understanding of the dump actions, and for long term measurement correlations.

As example, the "BTVDD**"** module analyses the Beam TV (BTV) images, taken on a screen placed a few meters in front of the TDE, to make sure that the beam reached the target at the expected position, and that the dilution was performed properly. The image is processed digitally to generate the skeleton of the beam sweep on the target. The skeleton position is then checked against the reference beam position computed based on the circulating beam filling pattern and average MKD and MKB kicker waveform.

The **"**BSRA" module checks that the beam population in the abort gap at the moment of the dump is below predefined limits.

INTEGRATION INTO THE LHC CONTROL SYSTEM

LHC Sequencer

Before injecting any beam into the LHC, the LHC Sequencer application executes a predefined list of tasks which will check the status of various devices and perform their initialisation. Checks include:

- The timestamp of the latest XPOC analysis session is compared to the timestamp of the latest LBDS trigger, to make sure that the latest dump action was properly analysed;
- The result of the latest XPOC session to make sure that no errors were detected during execution of the previous beam dump action.

If any of these sequencer tasks detects a problem regarding the XPOC analysis, the Engineer-In-Charge (EIC) is responsible for contacting an expert. The task that failed can be 'skipped' by the EIC, so it is not completely fail-safe; it is more informative.

LHC Software Interlock System

If an error is detected by the XPOC analysis server, the injection of beam in the LHC is inhibited using the Software Interlock System (SIS) by means of a so called "SIS latch". The SIS latch must be reset to proceed with beam operation. This is done using the XPOC-Viewer application, which provides two SIS-Latch reset buttons (visible on top of Figure 1), and is protected using the Role-Based Access Control (RBAC) mechanism [6]. The roles required to reset the XPOC SIS latch depend on the XPOC modules that failed. This is to make sure that the concerned expert has been contacted to check the problem before proceeding with injection in the LHC.

OPERATIONAL EXPERIENCES

The majority of XPOC analysis errors encountered so far were not related to LBDS problems, but to missing PM data buffers, mostly due to beam instrumentation equipment problems or CMW communication problems. Work is ongoing to have the very reliable data collection mechanism of the PMA framework implemented in all equipment used by the XPOC system. Nevertheless many interesting problems were detected by the XPOC system and some of them are briefly described below.

Problem with MKD Generators

MKD XPOC analysis errors occurred during operation with beam in 2008. After analysis of the long term trends of XPOC MKD logged analysis results, it appeared that some MKD characteristics were degrading. The problem was identified as a bad connection of a trigger cable [3].

Inject & Dump Synchronisation Error

During tests in Inject & Dump mode after a technical stop, the BTVDD XPOC analysis module failed at each dump action because the position of the pilot beam on BTVDD was much too high. The problem was traced back to a wrong LBDS control system parameter value for the Inject & Dump mode which was introduced during the technical stop. This resulted in the LBDS pulsing one LHC revolution too early (before beam injection), so the pilot beam (bunch #1) was not extracted at the beginning of the LBDS kicker pulses but at the end, and the extracted beam trajectory was not as expected. This problem was not seen by any other XPOC module, because all kickers pulsed properly, triggers were correctly synchronised with the beam revolution frequency, and no losses were detected by the BLM devices.

Losses Detected During Dump Action

When performing a beam dump, the BLM XPOC module often detects losses around protection equipment (absorbers and collimators). Recent analysis of the logged BLM and BSRA XPOC data showed a good correlation between the number of particles present in the beam abort gap at the time of dump and the losses measured on protection equipment. This correlation was the same for both beams. This is expected as all particles present in the abort gap during dump execution will be swept over all the LBDS protection equipment and the systems are identical for both beams. By combining these two measurements the XPOC system provided an easy way to confirm the rather complicated abort gap population measurement.

CONCLUSION

The XPOC system has proven its efficiency in detecting unexpected behaviour during execution of beam dump actions. It allowed the anticipation of some LBDS hardware problems before any system failure occurred, using the trends of logged XPOC analysis results. The LBDS stringent reliability specifications can only be met by the XPOC system guaranteeing that the latest dump was correctly executed, including all redundant parts. For this reason the reset of the XPOC SIS latch, which can only be done by experts having the required RBAC roles, is taken very seriously in daily operation of the LHC.

REFERENCES

[1] B. Goddard et al., "Initial results from beam commissioning of the LHC beam dump system", *Particle Accelerator Conference*, Vancouver, Canada, 2009.

[2] E. Carlier et al., "Design Aspect Related to the Reliability of the Control Architecture of the LHC Beam Dump Kicker Systems", *ICALEPCS*, Gyeongju, Korea, 2003.

[3] J. Uythoven et al., "Experience with the LHC beam dump post-operational checks system", *Particle Accelerator Conference*, Vancouver, Canada, 2009.

[4] M. Zerlauth et al., "The LHC Post Mortem Analysis Framework", *ICALEPCS*, Kobe, Japan, 2009.

[5] A. Antoine et al., "First Operational Experience of the LHC Trigger Synchronisation Unit", *ICALEPCS 2011*, Grenoble, France, 2011.

[6] K. Kostro et al., "Role-based authorization in equipment access at CERN", *ICALPECS*, Knoxville, Tennessee, USA, 2007.

THE RADIATION MONITORING SYSTEM FOR THE LHCb INNER TRACKER

O. Okhrimenko, V. Iakovenko, V. Pugatch, INR NASU, Kyiv, Ukraine
F. Alessio, G. Corti, CERN, Geneva, Switzerland

Abstract

The performance of the LHCb Radiation Monitoring System (RMS) designed to monitor radiation load on the Inner Trackersilicon micro-strip detectors is presented. The RMS comprises Metal Foil Detectors read-out by sensitive Charge Integrators. MFD is a radiation hard detector operating at high charged particle fluxes. RMS is used to monitor radiation load as well as relative luminosity of the LHCb experiment. The results obtained by the RMS during LHC operation in 2010–2011 are compared to the Monte-Carlo simulation.

THE LHCB DETECTOR AND THE SILICON TRACKER

The LHCb experiment is the forward spectrometer and one of the four experiments located at the LHC. The main aim of the LHCb is precise measurement of the CP-violation and research of the B-meson rare decays [1].

The LHCb, as high energy physics detector, consists of following parts: Vertex Locator (VELO), Inner and Trigger Trackers (IT, TT) and Outer Tracker to reconstruct tracks of charge particles and they decay vertexes and to separate Primary (proton-proton collisions) and Secondary (B-mesons decay) Vertexes (PV, SV); Magnet to measure charge particle momentum; Cherenkov Detectors (RICH1, RICH2) to separate kaons and pions; Hadronic and Electromagnet Calorimeters (HCAL, ECAL) to measure the particles energy; Muon detector to detect the muons.

The LHCb Silicon Tracker (ST) is a large-surface silicon microstrip detector that constitutes an important part of the LHCb tracking system. It uses single-sided silicon strip detectors with a strip pitch of approximately 200 µm, produced from 6" wafers and arranged into up to 38 cm long readout strips. he Silicon Tracker consists of two parts: the "Tracker Turicensis" is located in between RICH1 and the LHCb dipole magnet and the "Inner Tracker" [2] covers a cross-shaped area around the LHC beam pipe in tracking stations T1–T3, in between the LHCb dipole magnet and RICH2.

The level of charged hadron fluxes at the location of the silicon sensors of the IT-2 station varies from about 10^4 to $10^5\,\mathrm{cm^{-2}s^{-1}}$ at nominal LHCb luminosity ($2\times10^{32}\,\mathrm{cm^{-2}s^{-1}}$) [3]. These fluxes are high enough to make a significant damaging impact onto the performance of the IT sensors and their frond-end electronics (Beetle). So, IT requires the system to monitor radiation loads on Si-sensors. This is task for the Radiation Monitoring System.

THE RADIATION MONITORING SYSTEM FOR THE LHCB INNER TRACKER

The main goal of the Radiation Monitoring System (RMS) is a measurement of the radiation dose load onto silicon micro-strip sensors of the IT LHCb as well as front-end electronics in order to exclude their damage as a result of an unexpected radiation incident, i.e. change of the beam trajectory, partial beam loss in the region of the detector etc [4, 5].

The RMS is based on the Metal Foil Detector (MFD) technology. The principle of the MFD operation explores Secondary Electron Emission (SEE) phenomena from the metal foil surface (emission layer ∼ 10-50 nm) caused by impinging charge particles. SEE causes extra positive charge on an isolated metal foil read out by sensitive Charge Integrator (ChI).

Several MFD advantages have determined our choice for the IT RMS:

- The possibility to provide extremely low mass of the detecting material (from practical point of viewfew tens µm);
- Simple readout electronics (charge integrators and scalers);
- Low operating voltage (∼ 20 V);
- High radiation tolerance;
- Long term performance with minimal maintenance;
- Low cost.

From the technical point of view MFD is a 5-layer structure manufactured out of 50 µm thick Al foils supported by insulating epoxy frames. The central sensitive layer is connected to the readout electronics, while two neighboring (from both sides) accelerating layers are biased by positive voltage (HV, 24 V) to reduce recombination after SEE. The two outer shielding layers are grounded. RMS Sensor and accelerating layers are divided into 7 parts (110 × 75 mm, with a layout which is similar to IT silicon sensors size). The RMS consists of 4 modules (Top, Cryo, Bottom, Access) containing 7 sensors each (in total 28 sensors), which are located at IT-2 station (∼ 8.4 m from interaction point) around the Beam Pipe. Due to IT-boxes overlapping the Top-module is shifted up on ∼ 5 cm from the Beam Pipe.

The RMS readout electronics consists out of the six 5-channel sensitive ChIs [6] and 32-channels LVDS VME-scaler (C.A.E.N. V830 LC). The ChIs were developed at INR (Kyiv, Ukraine) and have been modified at MPIfK (Heidelberg, Germany). The ChI's principle of operation includes a current-to-frequency converter allowing to

achieve high dynamic range (up to 10^6). A current from the stable external source (250 pA) is injected to the ChI's inputs to make base lines (25 kHz). The tipical features of the RMS is presented in Table 1. The RMS is designed for the monitoring of charge particle fluxes exceeding ~ 2500 MIP/s per sensor.

Table 1: Typical Features of the RMS

Name	Value
ChI conversion factor	10 fA—1 Hz
SEE factor	~ 25 SE/MIP
RMS response	30 MIP/cm^{2}s—1 Hz

RESULTS

During the year 2010 LHC has provided colliding proton beams at 7 TeV (c.m. energy) delivering 42 pb^{-1} at LHCb, in total. The charged particle fluxes high enough to evoke a signal in the RMS were during high intensity beams (starting from the 20$^{\text{th}}$ of September'10) which contributed 29 pb^{-1} into the total delivered luminosity. As it is shown in Fig. 1 RMS response is linear correlated with the LHCb measured luminosity. This has been used to calibrate the RMS for measuring luminosity in future as well as to extrapolate RMS measured data on low intensity beams colliding which usually occur at the beginning of the LHC operation after shutdowns.

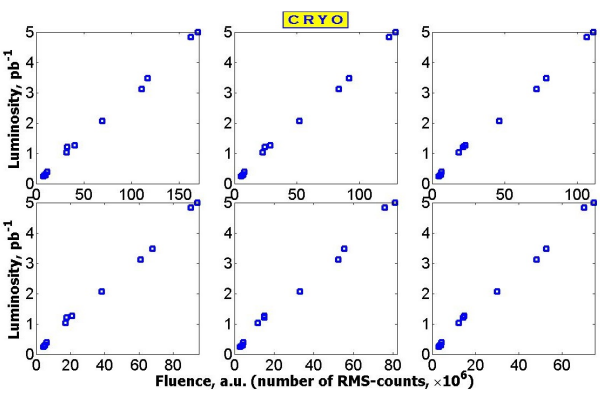


Figure 1: Correlation between the RMS response and the LHCb measured Integrated Luminosity in 2010. The response from 6 sensors of the RMS's Cryo module is presented for illustration.

LHCb Integrated Luminosity Measured by the RMS

In 2011, the first protons collisions at LHCb occurred on the 14$^{\text{th}}$ of March. Till 18$^{\text{th}}$ of March low intensity beams collided producing charged particle fluxes insufficient to evoke a response in the RMS. Total integrated luminosity of these first collisions has not exceeded 3% of the annual one.

Due to power cut the RMS had not been operating 10 days in July missing 41 pb^{-1} of data which correspond to $\sim 6\%$ of annual integrated luminosity.

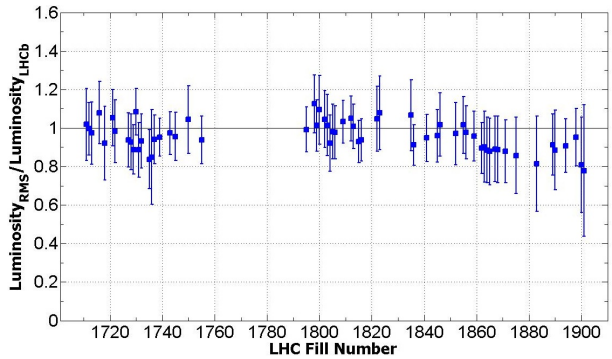


Figure 2: A Ratio of the luminosity measured by RMS to the LHCb one.

So, the RMS measured about 90% of the total integrated luminosity. The uncertainty of those measurements is about 10%. This accuracy is sufficient for the on-line monitoring of the integrated luminosity during pp-collisions at the LHCb.

The RMS data agrees well with the LHCb ones [7]. Figure 2 presents a ratio of the integrated luminosity measured by RMS to LHCb's one calculated for the 61 LHC Fills (~ 400 pb^{-1}, April-May'11).The uncertainty of the on-line LHCb's luminosity data is about 10%.

Radiation Load over IT Si-sensors Measured by the RMS

As it was mentioned above, RMS's main goal is monitoring of the radiation load on LHCb Inner Tracker sensors. During the 2011 year RMS performance allowed to measure 90% of the radiation load. Perfect linearity of the RMS response with respect to the integrated luminosity (see Fig. 1) makes it possible to extrapolate data into the 'unmeasured zone.

In total, over 1 fb^{-1} integrated luminosity has been delivered to the LHCb in 2011 year. Dose Distribution over IT Si-sensors measured by the RMS corresponding to this integrated luminosity is shown in Fig. 3.

An absorbed dose varies from 100 to 400 Gy depending upon the sensor. Sensors closest to Beam Pipe get higher doses then peripheral ones. These doses correspond to $(0.4\text{–}1.5) \times 10^{12}$ MIP/cm^2 which results in 50–200 µA leakage currents increase over Si-sensors, respectively. This requires Si-sensors cooling down and bias voltage tuning to keep reliable operation of the IT. The uncertainty of the measurements does not exceed 10%.

The leakage currents data evaluated by the RMS are in good agreement with a direct leakage current measurements as well as with the Monte-Carlo predictions for the charged particle fluxes (see below).

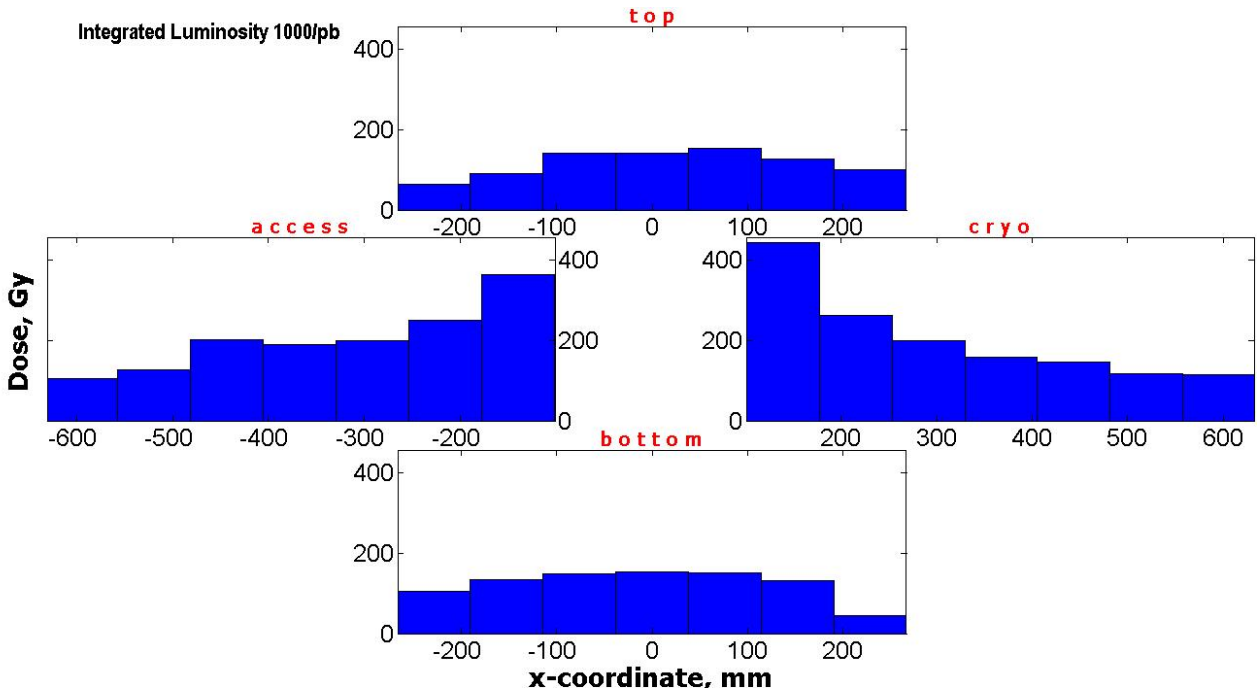


Figure 3: Dose Distribution over the IT Si-sensors measured by the RMS in 2011.

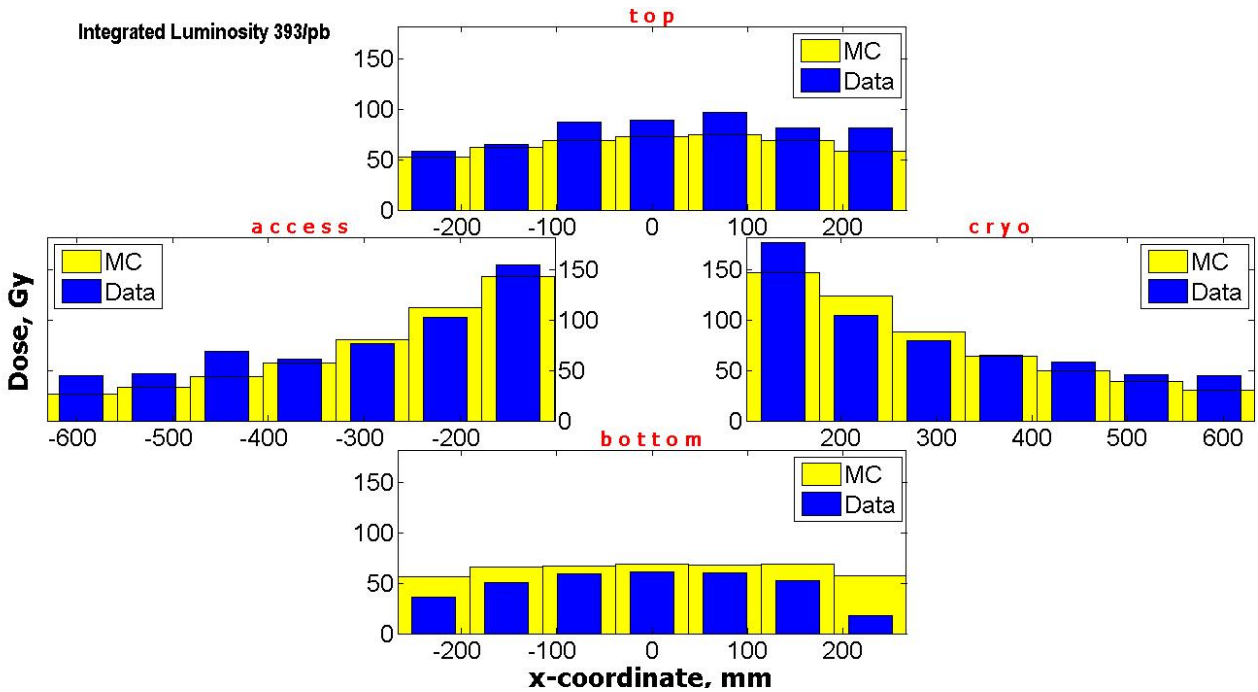


Figure 4: Comparison between real and MC simulated RMS data.

Comparison Between Real and Monte-Carlo Simulated RMS Response

Using standard LHCb software (Gauss v38r9) [8], 10k events under following condition were generated:

- a center of mass energy of the colliding proton beams—7 TeV,
- the average number of pp-interaction (including elastic which is not visible in the detector) per bunch crossing (ν) is 2.5,
- different LHCb Dipole Magnet polarity—up and down,
- charge particle, only, were included.

These conditions are very similar to the real one during 2011 LHC operational year.

The resulting particle fluxes distribution over sensors in comparison with the data measured by the RMS is shown in Fig. 4.

Good agreement is observed for all modules but the 'Bottom' one. This disagreement is under study and might be caused by the Beam Pipe supporting tools located just in front of the RMS not included in the MC simulations.

CONCLUSION

During the LHC operational year 2011 the Radiation Monitoring System has provided monitoring of the radiation load on Si-sensors of the LHCb Inner Tracker. RMS data have allowed also to determine the integrated luminosity as well. The RMS data are in good agreement with data obtained by other detectors as well as with Monte-Carlo simulations. These data are planned to be included into the on-line monitoring of the radiation load and integrated luminosity.

ACKNOWLEDGMENTS

We would like to thank Silicon Tracker and Beam & Background groups and the LHCb Collaboration for exciting studies this year. Special thanks to F. Blanc, H. Voss, J. van Tilburg, M. Needham and R. Jacobsson.

REFERENCES

[1] The LHCb Collaboration, JINST S08005 (2008).

[2] The LHCb Collaboration, LHCb Inner Tracker Technical Design Report, CERN/LHCC 2002-29.

[3] V. Talanov, Radiation Environment at the LHCb Inner Tracker Area, LHCb Note 2000-013.

[4] V. Pugatch et al., Radiation Monitoring System for the LHCb Inner Tracker, LHCb Note 2007-062.

[5] V. Pugatch et al., Ukr. J. Phys, 54(4) (2009) 418.

[6] V. Kyva, N. Tkatch, Scientific Papers of the Institute for Nuclear Research, 2(4) (2001) 72.

[7] https://lbweb.cern.ch/groups/online/OperationsPlots/OperationsDashboard.htm

[8] http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/gauss/

EQUIPMENT AND MACHINE PROTECTION SYSTEMS FOR THE FERMI@Elettra FEL FACILITY*

F. Giacuzzo, L. Battistello, L. Fröhlich, G. Gaio, M. Lonza, G. Scalamera, G. Strangolino, D. Vittor, Sincrotrone Trieste, Trieste, Italy.

Abstract

FERMI@Elettra is a Free Electron Laser (FEL) based on a 1.5 GeV linac presently under commissioning in Trieste, Italy [1]. Three PLC-based systems communicating to each other assure the protection of machine devices and equipment. The first is the interlock system for the linac RF plants; the second is dedicated to the protection of vacuum devices and magnets; the third is in charge of protecting various machine components from radiation damage. They all make use of a distributed architecture based on fieldbus technology and communicate with the control system via Ethernet interfaces and dedicated Tango device servers. A complete set of tools including graphical panels, logging and archiving systems are used to monitor the systems from the control room.

INTRODUCTION

The protection systems are based on Siemens S7 PLCs with an extensive use of Profibus to connect several distributed I/O peripherals and LCD operator panels. The communication between systems is realized either by means of Profibus or digital I/O signals, while Ethernet-TCP/IP is employed to interface to the FERMI@Elettra control system [2] using the Send/Receive protocol and dedicated Tango servers.

In all, the protection systems make use of five 315-2DP and 16 IM151 CPUs, 30 operator panels and 31 Profibus nodes. The systems manage in total about 1900 digital inputs, 500 digital outputs and 250 analog inputs.

LINAC RF PLANTS INTERLOCK SYSTEM

As shown in Fig. 1 each RF modulator is equipped with one PLC (CPU_MU), which guarantees the required performance in terms of reaction time. The goal is to allow no more than one linac shot after an interlock alarm is detected; given that the linac maximum repetition frequency is 50Hz, the protection action must be accomplished in less than 20 ms. This has also been achieved with an accurate design of the software architecture and a thorough programming. It has been necessary, for example, to avoid pre-compiled functions in favour of very primitive home-made functions and, whenever possible, to make extensive use of "jump" instructions. With a Siemens IM151 CPU controlling about 18 digital I/Os and eight analog inputs, the maximum reaction time is 12 ms.

Each RF plant has one touch-screen operator panel manufactured by UNIoP. A number of synoptic panels display the modulator interlock states and the analog input values, such as temperatures and klystron focalization currents, and allow the operator to set the corresponding interlock values.

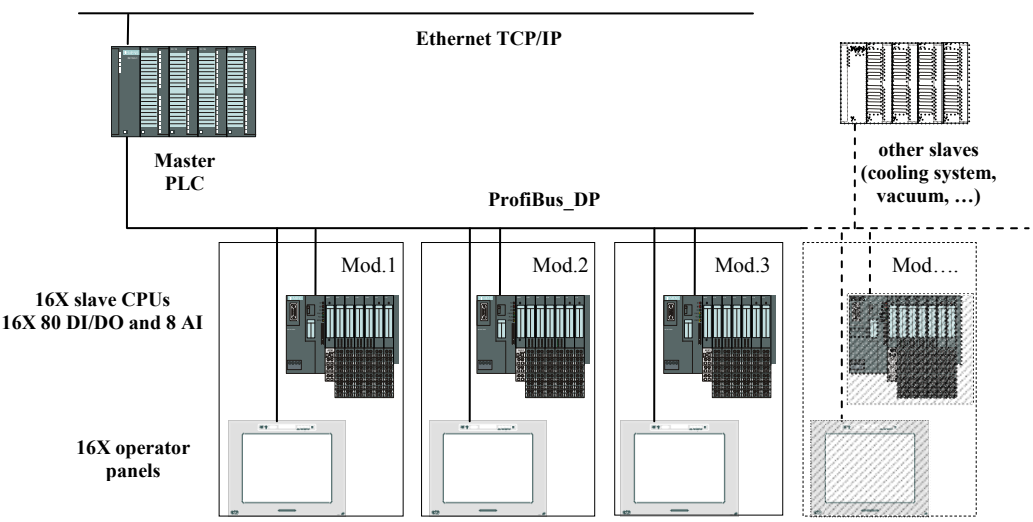


Figure 1: Block diagram of the linac RF plants interlock system.

* This work was supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3.

An example of panel representing the logics of the signals that enable the various RF plant subsystems is shown in Fig. 2.

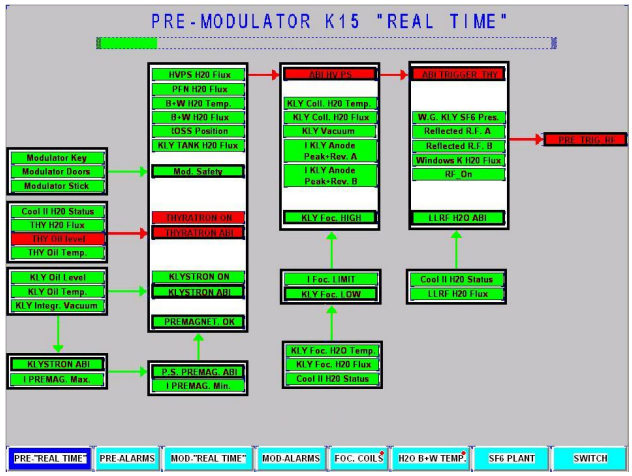


Figure 2: A page of the RF interlock operator panel.

All the CPU_MUs and the operator panels are connected through Profibus to a "Master PLC" configured as Profibus master, which does the following:

- Distribute to the CPU_MUs some common interlock alarms related for example to the general water cooling system, which don't need a short reaction time;
- Collect all the data from the CPU_MUs and send them through Ethernet to the Tango server; the same data are also sent to the local operator panels.
- Receive control commands from the Tango server and send them to the proper CPU_MU.

With this architecture the interlock system of each RF plant is independent from each other. If one of them fails the others can continue working. Within some functional limitations the systems can also work without the "master PLC".

During the first period of the machine commissioning, we suffered from a few interruptions of the communication between the master and slaves probably due to electromagnetic noise from klystron modulators. The problem has been solved by modifying the default DP Profibus profile; in particular, we have increased from one to ten the number of connection attempts from master to slave.

MAGNETS AND VACUUM INTERLOCK

The interlock for the protection of the accelerator electromagnets switches off the power supply in case of alarm from the cooling system of the corresponding magnet. Depending on the magnet type, alarms can be generated by flux-meters and thermo-switches. For safety reasons a lamp driven by the interlock system in placed on the magnets with potentially dangerous voltages. Moreover, the interlock system is also used to safely inhibit the power supplies in case of personnel access in the tunnel.

The purpose of the vacuum interlock is to avoid the propagation of possible leaks along the vacuum chamber. In the accelerator the PLC receives vacuum alarms from ion pump and vacuum gauge controllers by means of voltage-free contacts; the alarm thresholds (set points) are set on the controllers. In the machine front-end, instead, also analogic signals are acquired from temperature sensors, turbo-ionic pumps and vacuum gauges.

According to the coded protection logics, the PLC closes at least two valves in order to isolate the segment of vacuum chamber where an anomalous pressure increase has been detected; at the same time the PLC disables the electron beam to avoid damages of the valves. The vacuum valves can be remotely controlled (status reading and open/close command) either by using the local operator panels (Fig. 3) or via the Ethernet TCP/IP interface.

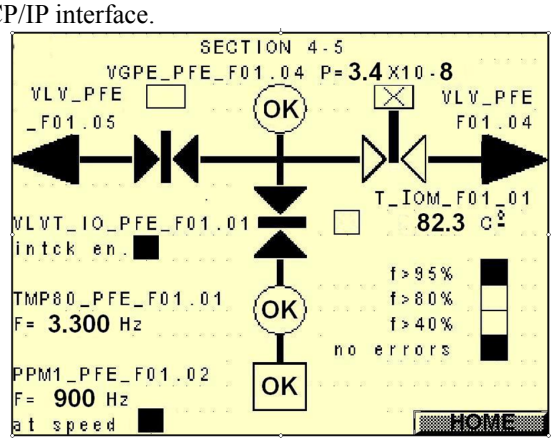


Figure 3: An operator panel display dedicated to the frontend vacuum interlock system.

MACHINE PROTECTION SYSTEM

The Machine Protection System (MPS) protects the FEL undulators from the deposition of excessive radiation doses caused, for example, by bad alignment of the electron beam. Several diagnostics are used to detect radiation, including ionization chambers, charge loss monitors and Cherenkov fibres [3]. The block diagram of the MPS is shown in Fig. 4.

In order to guarantee the required reaction time, a specific PLC has been dedicated to the MPS. It directly receives the alarm signals from the ionization chambers electronics. The analog signals generated by charge loss monitors and Cherenkov fibres are acquired by a VME system running Linux and the Xenomai real-time extension [4], which performs the signal processing and generates the alarms. The connection of the VME system with the PLC is realized by means of digital outputs and a heart-beat signal that synchronizes the communication.

Other signals not needing a fast response are collected by the PLC via Profibus from the interlock system.

In case of alarm, the PLC closes a shutter that stops the laser of the photo-injector, thus switching off the electron beam. The measured reaction time is less than 10 ms, so only one shot could eventually be fired after the alarm is received.

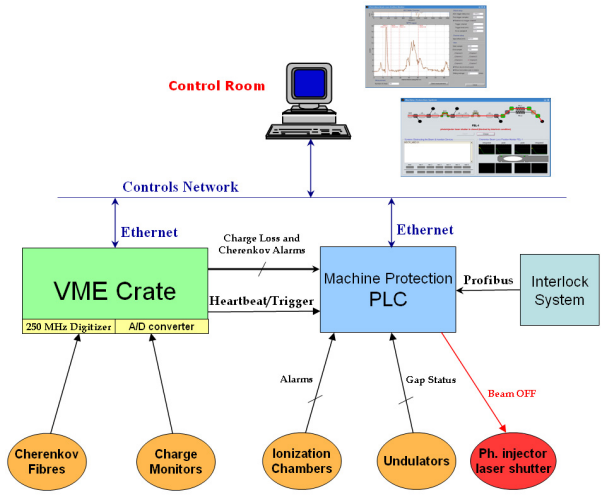


Figure 4: Block diagram of the Machine Protection System.

Since incorrectly inserted multi-screens hit by the electron beam can cause the release of huge radiation fields, the screens position is detected by means of micro-switches and the beam is disabled when they are moving. When a given screen arrives in the final position, the beam is automatically enabled only if that screen can be safely hit by electrons.

The reading of the undulators gap status (open/closed) is used to stop the beam only if the alarm is coming from a location were the undulator gap is closed. Moreover, the state of the bending magnets power supplies are used to determine the real path of the electron beam, so that only screens that can actually be reached by the electron beam are taken into account. A future upgrade of the system will provide analog readings of the bending currents.

In order to allow machine operations in the presence of anomalous situations during the machine commissioning, a feature has been implemented to selectively "bypass" some alarms. This possibility is only permitted to expert people.

The PLC cycle is synchronized with the linac trigger (up to 50Hz). At each linac shot the PLC compile a portion of a Data Block (DB) containing all the alarm states and adds to it a time stamp. When the DB is fully compiled with 50 shots, it is sent via Ethernet to a Tango server which stores it into a database. With this feature every event detected by the MPS is recorded and made available for analysis.

THE SUPERVISION SYSTEMS

Supervision applications run in the FERMI@Elettra control systems and are developed using the Tango framework. The interlock systems communicate with the control system through Ethernet links and dedicated Tango servers, which acquire alarms and send commands.

At every cycle each PLC compiles a DB with the data acquired from the field, both digital (true/false) and analog (floating point) values. If the PLC detects the variation of at least one of the values with respect to the previous cycle, it adds a time-stamp and sends out the DB via the Ethernet port. This is called "Real-Time DB".

In order to detect and memorize fast events that normally could be lost, a second DB called "Alarm DB" similar to the first one, latches the active alarms until an acknowledge command is received via the external interface. Also the Alarm DB is sent on variation.

A Tango Device Server for each PLC is in charge of receiving the DBs and saving them into a MySQL database. A second level of Tango servers communicating with the first level has been developed to extract data from the DBs and export every single value as a meaningful Tango attribute. The servers also receive commands from Tango client applications and forward them to the PLCs.

A number of graphical interfaces have been developed using Matlab and QTango, a C/C++ graphical library for the Tango control system. They display the status of the systems and warn operators of interlock alarms. Fig. 5 is an example of a synoptic panel representing the status of the vacuum interlock.

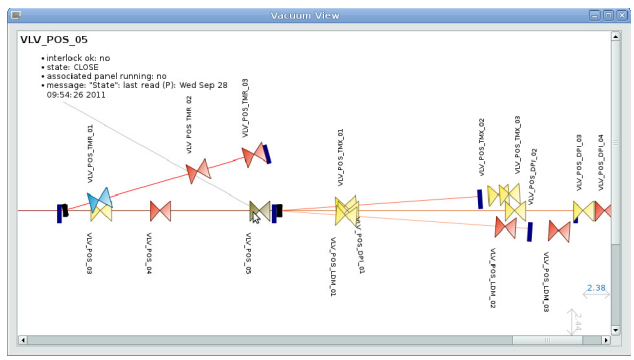


Figure 5: Synoptic panel of the vacuum interlock system.

REFERENCES

[1] S. Di Mitri, "Commissioning and Initial Operation of FERMI@Elettra", IPAC 2011, San Sebastián, September 2011.

[2] M. Lonza et al., "The Control System of the FERMI@Elettra Free Electron Laser", ICALEPCS2009, Kobe, October 2009.

[3] L. Fröhlich, A. I. Bogani, K. Casarin, et al. Instrumentation for machine protection at FERMI@Elettra. Proc. DIPAC'11, Hamburg, Germany, May 2011.

[4] L. Pivetta et al., "The FERMI@Elettra distributed real-time framework", these proceedings.

PROTECTING DETECTORS IN ALICE

Mateusz Lechman[1], Andre Augustinus[1], Peter Chochula[1], Antonio Di Mauro[1], Lennart Stig Jirden[1], Peter Rosinsky[1], Heinrich Schindler[1] Giacinto De Cataldo[1, 2], Ombretta Pinazza[1, 3], Alexander Kurepin[1, 4], Alberto Moreno[5]
[1]CERN, Geneva, Switzerland
[2]INFN-Bari, Bari, Italy
[3]INFN-Bologna, Bologna, Italy
[4]RAS/INR, Moscow, Russia
[5]Universidad Politécnica de Madrid, Madrid, Spain

Abstract

ALICE (A Large Ion Collider Experiment) is one of the big LHC (Large Hadron Collider) experiments at CERN in Geneva. It is composed of many sophisticated and complex detectors mounted very compactly around the beam pipe. Each detector is a unique masterpiece of design, engineering and construction and any damage to it could stop the experiment for months or even for years. It is therefore essential that the detectors are protected from any danger and this is one very important role of the Detector Control System (DCS). One of the main dangers for the detectors is the particle beam itself. Since the detectors are designed to be extremely sensitive to particles they are also vulnerable to any excess of beam conditions provided by the LHC accelerator. The beam protection consists of a combination of hardware interlocks and control software and this paper will describe how this is implemented and handled in ALICE. Tools have also been developed to support operators and shift leaders in the decision making related to beam safety. The gained experiences and conclusions from the individual safety projects are also presented.

INTRODUCTION

The Large Hadron Collider (LHC) at CERN is the most powerful accelerator ever built. A top energy of 360 MJ will be stored when two counter-rotating proton beams, containing up to 2808 bunches of 1.1 x 10^{11} protons, will collide every 25 ns at a centre of mass energy of 14 TeV in each of the four experiments distributed along the 27 km accelerator ring.

Local beam losses at various levels may induce a quench of the super-conducting magnets or even a physical damage to machine components and experiments detectors. Such losses can be the consequence of various instabilities (beam failures) happening during beam injections or with circulating beams.

Depending on the nature and amount of beam loss, several kinds of damages can be produced: high voltage trips and accelerated ageing in gaseous detectors, single event upsets in front-end electronics or even permanent damage to semiconductor devices and electronics components.

In order to prevent such damages a complex protection system has been developed comprising numerous diagnostic systems and passive protection elements [1]. The ALICE experiment [2] is protected by a dedicated Beam Condition Monitor (BCM) system connected to the LHC Beam Interlock System (BIS) [3]. In addition safe states have been defined for all sub-detectors to establish the best conditions implying the smaller risk of damage in given beam operating modes.

BEAM CONDITION MONITOR

The Beam Condition Monitor system provides real-time monitoring of the radiation level in the ALICE experimental area. In order to protect the detectors against multi-turn beam failures it triggers a beam dump if the instantaneous radiation level crosses a certain threshold.

Used Components

The ALICE BCM design was based on the Beam Condition Monitor system of the LHCb experiment [4].

The beam background is measured using pCVD (polycrystalline Chemical Vapour Deposition) diamond sensors that are grouped in two stations located close to the beam pipe on both sides of the interaction point. The readings from these sensors are digitized by CFC (current-to-frequency converter) cards and then sent to a TELL1 board [5] via optical links. The TELL1 board calculates running sums (RS) over 1, 2 and 32 CFC data frames with corresponding integration times of 40 µs (RS1), 80 µs (RS2) and 1.28 ms (RS32), respectively. In addition "RS32 Sums" are calculated by summing up the RS32 values of 5 out of 8 sensors of one station discarding the two highest values and the lowest value.

Access to the registers of the TELL1 board is provided via an on-board credit card-sized PC (CCPC). The CCPC can communicate via DIM (Distributed Information Management System) with the SCADA software package PVSS. The DIM server is running on the CCPC and a DIM client is part of the PVSS project.

Interface to BIS

ALICE provides three flags (user permits) to the LHC Beam Interlock System via the TELL1 board:

- Beam permit - inhibiting/allowing the beam circulation (acting on both beams simultaneously),
- Injection permit for beam 1 – injected via the TI2 transfer line from SPS (Super Proton Synchrotron) to LHC,
- Injection permit for beam 2.

A beam dump is triggered by the TELL1 firmware via removal of the beam permit if values of monitored running sums exceed defined thresholds.

The injection permits are controlled by the central Detector Control System (DCS) [6] and they are set when ALICE is prepared for incoming beam.

Control Layer

A dedicated PVSS project is used to supervise and control the performance of the TELL1 board and auxiliary devices such as CAEN power supply and VME crate. The designed panels provide experts and DCS operators with all important information including status of BIS flags, the beam background level and reasons of beam aborts.

In case of emergency, the control and monitoring of the CCPC can be continued by the DCS operator (even without the PVSS layer and DIM) via emergency command line scripts, based on the SSH connection tool PLINK.

The values of the BCM thresholds are changed dynamically depending on the state of the most sensitive ALICE sub-detectors and a "beam mode" published by the CCC (CERN Control Centre). The more strict limits are activated before turning on the selected subsystems.

Post Mortem

The causes of emergency beam aborts have to be revealed to improve protection systems and operational procedures. In such cases, analysis of data registered shortly before the dump is essential.

The incoming data stream from the CFC cards is written into a circular buffer on the TELL1 board. This buffer is frozen when a Post Mortem (PM) signal is received via the LHC machine timing system.

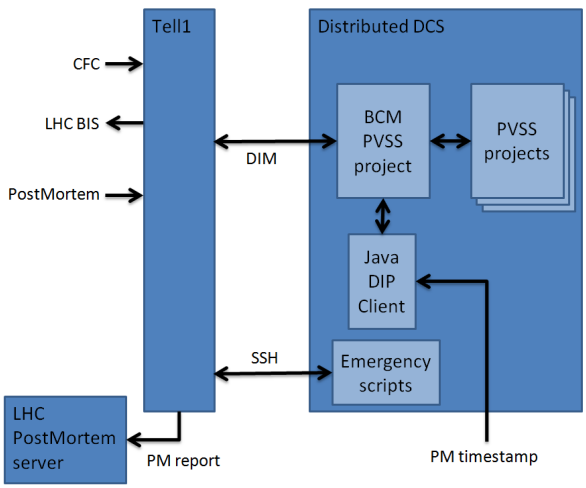


Figure 1: Beam Condition Monitor - the control layer.

The content of the circular memory buffer is marked using the published timestamp of the latest PM event and then sent to the LHC central PM server, from where it can be further analyzed.

The timestamp is distributed over Data Interchange Protocol (DIP) in nanoseconds. Since the PVSS DIM client does not support such accurate time resolution, a specialized client was implemented in Java for retrieving it.

The main communication flow in the BCM system is presented in Fig. 1.

OPERATING THE DETECTORS

ALICE consists of many subsystems integrated via a distributed PVSS project. The DCS shifter operates through a central User Interface (UI), which is connected to all sub-detectors and to the common systems like electricity, environment monitoring etc. The User Interface collects all the system states and alerts, and is able to send commands to change the status of the detectors.

The UI panels are designed with the goal of simplifying the actions and minimize the time and risk of mistakes. The most frequent procedures have been coded into scripts and panels, and the operator is asked to activate an existing procedure when needed, instead of interacting with the single detectors following their different and dynamic requirements

The SAFE and SUPERSAFE States

Certain LHC operations require detectors to be in a compatible, so called SAFE condition. During the LHC Injection, TI2 setup or beam adjust, the sensitive detectors typically lower their voltages to protect the electronics against excessive charge deposits.

During certain operations (such as testing of new LHC filling schema etc.) the detectors put even more restrictions on their settings. This condition is called SUPERSAFE.

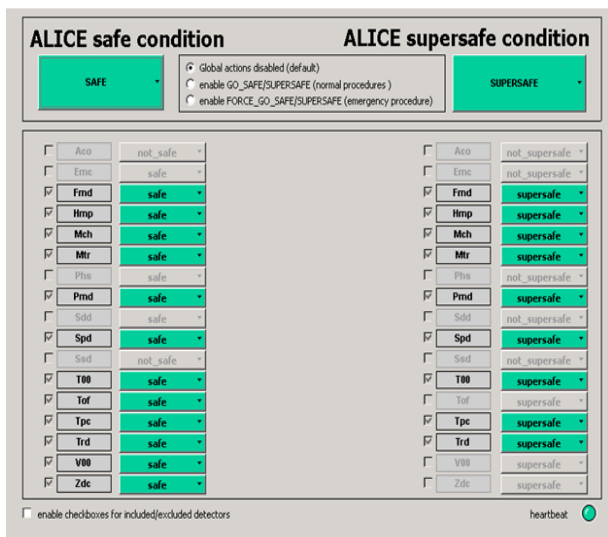


Figure 2: The SAFE/SUPERSAFE panel.

Each detector declares if it meets the SAFE and SUPERSAFE conditions. The central system collects this information and sends to the subsystems the GO_SAFE or GO_SUPERSAFE command through a PVSS datapoint to obtain the required level of safety. If a detector is in a SUPERSAFE condition, it can be brought

back to SAFE using the RESTORE_SAFE action. The commands are available to the DCS operator from the SAFE/SUPERSAFE panel presented in Fig. 2.

The shifter can also send global commands (GO_SAFE, GO_SUPERSAFE and RESTORE_SAFE) that trigger relevant actions on side of all the detectors. When all the detectors are SAFE (or SUPERSAFE), the global state of ALICE is declared as SAFE (or SUPERSAFE).

An exclusion matrix is introduced to temporarily ignore individual detectors from the calculation of the global state. This matrix can only be changed under the supervision of the shift leader and the detector expert. The exclusion/inclusion from/to the safety matrix is also set via the SAFE/SUPERSAFE panel.

The GO_READY Matrix

When the beam condition requiring the SAFE/SUPERSAFE state is not present anymore, the detectors are sent, on request of the shift leader, to the READY state via the GO_READY command to start the data taking.

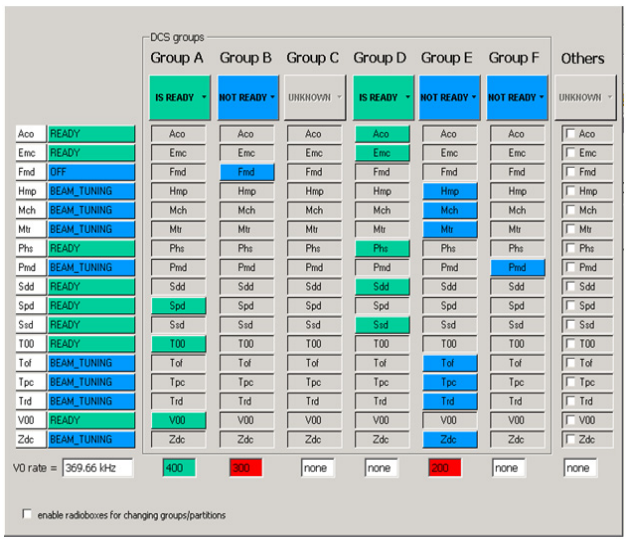


Figure 3: The GO_READY panel.

Different detectors will switch on the HV in different phases of the beam operation (the so-called luminometers right after injection, the gaseous detectors only at stable beams).

In order to minimize the risk of a mistake of the DCS operator and the time needed to send to READY the relevant detectors, a panel has been prepared which groups the subsystems and allows sending the GO_READY command in parallel to the different groups (Fig. 3).

The V0 Rates

The risk of HV trips and potential sub-detectors damage can be estimated using the V0 subsystem. This detector is based on two disks of segmented plastic scintillator (76 cm in diameter) located asymmetrically with respect to the ALICE interaction point (+340 cm, -90 cm).

Due to its enhanced timing performance V0 can detect both beam-beam and beam-gas collisions, thus providing separate numbers of collisions (rates) for luminosity and background. The sum of these rates lets the experiment monitor the amount of charged particles crossing the sub-detectors.

Right after the beam injection and the energy ramp, ALICE and the LHC usually agree on a desired luminosity level. If the sum of luminosity and the background rates is above a fixed threshold for some sensitive gaseous detectors then they should not be sent to READY until the V0 rates decrease below the threshold.

A control is then implemented in the GO_READY panel to alert the DCS shifter not to send to READY the detectors until this condition is valid. The decision to continue or abort the GO_READY command is then left to the shift leader.

If the V0 rates are too high and the detectors are READY, a severe alert is raised and again the SL takes the responsibility to ramp down the different sub-detectors.

THE HANDSHAKE MECHANISM

The Handshake is a communication protocol between the CERN Control Centre (CCC) and the LHC experiments. Its purpose is to inform the shift crews of the experiments about any beam activity to be performed by the machine. Another goal of the Handshake is to obtain permission from the experiments before starting actions like injection, adjust or beam dump. After receiving the notification about the beam activity, the experiments should take necessary actions to prepare the detectors for this activity, to avoid damages that the beam may cause.

The Handshake mechanism is based on the exchange of messages (“LHC commands” and “ALICE responses”) between the CCC and ALICE via DIP. On the ALICE side, the logic of the Handshake is implemented as a finite state machine using SMI scripts from the fwFSM component developed within the JCOP project.

Injection Handshakes

From the point of view of safety, the most critical are Injection Handshakes. Their main purpose is to make all the subsystems SAFE (and if required SUPERSAFE) and to remove the injection interlock. The relations between different phases of the Injection Handshake and protection mechanisms are illustrated in Fig. 5.

When the CCC is ready to inject the beam they publish the “WARNING” message. The ALICE DCS operator, using dedicated control panel (Fig. 4), confirms it by executing the “INJECT_PREPARE” action and then starts to prepare for the injection. During the next 5 minutes the ALICE experiment should be moved to the global SAFE (or SUPERSAFE) state. The operator executes the “CONFIRM_INJECT” action when the experiment is prepared. This command also sends a request to the BCM system to grant the injection permits.

After all the experiments report they are ready the CCC sends the "READY" command and only then the beams can be injected into the LHC. When the injection is completed the CCC publishes the "OK" message, which triggers removal of the injection permits. Starting from this moment ALICE no longer needs to be in the SAFE/SUPERSAFE state.

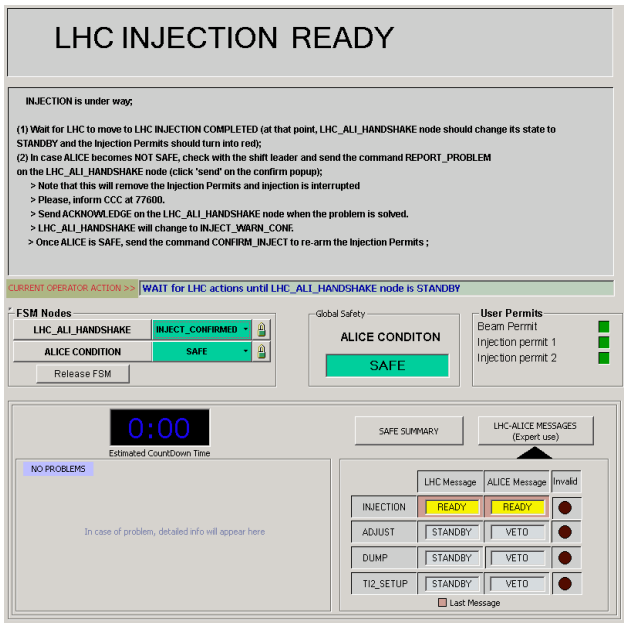


Figure 4: The Handshake control panel.

Emergency Situations

The handshake protocol can also handle exceptional situations, like violation of the SAFE state or a communication problem in DIP. In such situations the state of handshake is changed automatically to emergency state "PROBLEM_REPORTED" in which the injection permits are not granted. This state can also be set manually by the DCS operator upon the shift leader request in justified cases.

CONCLUSIONS

The main task of the central DCS team was to design a system to keep detectors in the safest condition during the most dangerous machine modes (from injection up to adjust) and a monitoring system able to react to anomalous beam conditions. These goals were achieved and the experience gained during 1.5 years of operation has been used to better define the safe states of each detector and values of the BCM dump thresholds.

At the moment, the most dangerous for ALICE are fast beam losses produced during injection by kicker magnet failures since the reaction of the BCM and LHC Beam Loss Monitor systems is too late. Therefore, the proper definitions of the SAFE and SUPERSAFE states are essential. Wrong definitions can lead to serious incidents, like during big beam losses in July 2011, when part of a calibration system of Silicon Drift Detector was damaged and the safe policy had to be reviewed for this subsystem.

On the other hand slow losses related to circulating beams can be detected in due time and beams can be dumped typically within 2-3 orbits before any damage is produced.

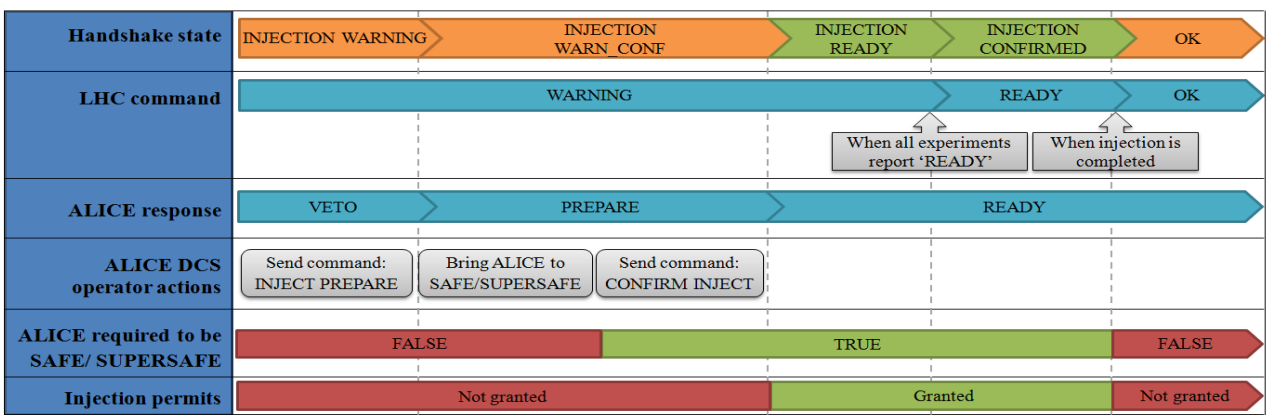


Figure 5: The Injection Handshakes procedure in ALICE.

REFERENCES

[1] R. Schmidt et al., "Protection of the CERN Large Hadron Collider", New Journal of Physics 8 (2006), 290.

[2] The ALICE Collaboration et al., "The ALICE experiment at the CERN LHC", JINST 3, S08002, 2008.

[3] B. Todd, A. Dinius, P. Nouchi, B. Puccio, R. Schmidt, "The Architecture, Design and Realisation of the LHC Beam Interlock System", Proceedings of the 10th ICALEPCS, Geneva, 2005.

[4] M. Domke, C. Ilgnerl, S. Kostner M. Lieng, M. Nedosl, S. Sauerbreyl, S. Schleich, B. Spaanl, and K. Wardal, "Commissioning of the Beam Conditions Monitor of the LHCb Experiment at CERN", 2008 IEEE Nuclear Science Symposium Conference Record, p 3306 – 3307.

[5] G. Haefeli, A. Bay, F. Legger, L. Locatelli, J. Christiansen, and D. Wiedner, "TELL1 Specification for a common read out board for LHCb", Technical Report LHCb 2003-007, 2005.

[6] P. Chochula, L. Jirden, A. Augustinus, G. de Cataldo, C. Torcato, P. Rosinsky, L. Wallet, M. Boccioli, L. Cardoso, "The ALICE Detector Control System", IEEE Transactions on Nuclear Science, volume 57, 472 - 478, 2010.

DEVELOPMENT STATUS OF PERSONNEL PROTECTION SYSTEM FOR THE IFMIF/EVEDA ACCELERATOR

T. Kojima*, T. Narita, K. Tsutsumi, K. Nishiyama, H. Takahashi, H. Sakaki
Japan Atomic Energy Agency, IFMIF Accelerator Development Group,
2-166 Obuchi, Omotedate, Rokkasho-mura, Kamikita-gun, Aomori 039-3212, Japan

Abstract

In the IFMIF/EVEDA Project, the prototype accelerator experiments to produce a deuteron (D^+) beam with the beam current of 125mA accelerated up to 9MeV at the CW mode are planned. The Personnel Protection System (PPS), which has two functions of radiation safety management and accelerator operation management, has to protect the personnel against unnecessary exposure, electrical shock hazard and the other dangerous phenomena. Since the radiation safety management during and after beam operation is a critical issue for the IFMIF/EVEDA accelerator, PPS with high reliability is indispensable. To realize the high reliability, for PPS design policy, we decided to use and customize proven systems and hardware, and applying dual PLC system [1]. This article presents the PPS design for the radiation safety management.

INTRODUCTION

The International Fusion Materials Irradiation Facility / Engineering Validation and Engineering Design Activities (IFMIF/EVEDA) Project is carried out at the BA site in Rokkasho, Aomori, Japan. The prototype accelerator consists of Injector extracting D^+ beam energy of 100keV, a 175MHz RFQ (4-vane type; output beam energy up to 5MeV), the first section of superconductive linac (SRF Linac; output beam energy up to 9MeV), beam dump, and the other accelerator subsystems [2].

Personnel Protection System (PPS) is a part of the control system for the IFMIF/EVEDA prototype accelerator. The PPS has two functions, which are radiation safety management and accelerator operation management in order to protect the personnel against unnecessary exposure, electrical shock hazard and the other dangerous phenomena.

Because the D^+ beam of 125mA/9MeV at the CW mode operation will be produced for the engineering validation, the radiation safety management is very important since the high radiation environment in the accelerator vault is assumed during and after beam operation. Therefore, the development of PPS with high reliability is indispensable for the radiation safety management for the personnel entering/leaving of radiation controlled area. In order to obtain the high reliability, we decided to use and customize the proven system and hardware to eliminate risks by initial failure, and apply dual PLC system to prevent malfunction.

*kojima.toshiyuki@jaea.go.jp

CONFIGURATION OF PPS

Fig. 1 shows a general configuration of the PPS. The PPS consists of a couple of PLCs, the PPS console, personal key boxes, door control boxes, emergency stop buttons, cameras for monitoring etc. In order to centralize status of the PPS equipments, all signal paths are hardwired from equipments to the PLCs.

One PLC system consists of one set of PLC and signal path which is composed to cables, limit switches and so on (for example, PLC-A and signal path for PLC-A in Fig. 1). The PPS consists of PLC-A system and PLC-B system and each system is configured separately and works independently.

The dual PLC system is adopted for higher reliability because the PPS only permit some actions when both PLC systems make same judgement. For example, workers can only operate the accelerator subsystems when both PLC-A and PLC-B system permit the operation.

The PPS console is used for monitoring and operating at the control room. The personnel entering/leaving of the radiation controlled area is managed by taking/returning personal keys. Access doors on the controlled area can be opened by the personal keys.

In the accelerator vault etc., some emergency stop buttons are installed. The beam operation is stopped by pushing the button.

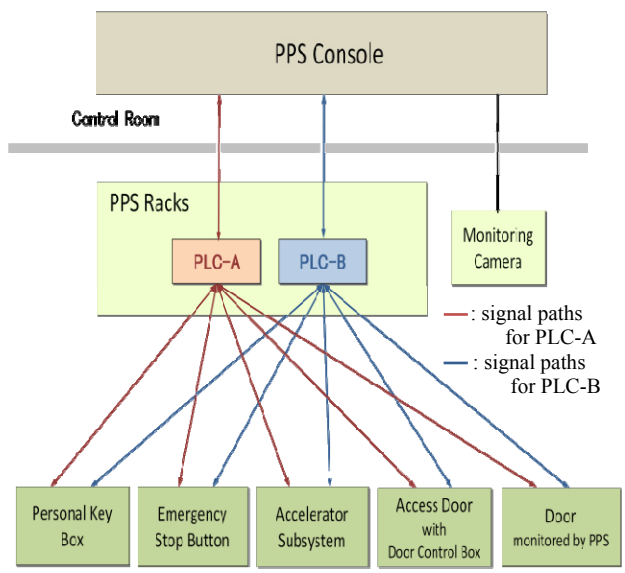


Figure 1: General configuration of PPS for the IFMIF/EVEDA prototype accelerator.

RADIATION CONTROLLED AREA

Fig. 2 shows the radiation controlled area of the PPS and the layout of main equipment for the PPS. The controlled area consists of the accelerator vault (yellow part in Fig. 2), radiation hot area (blue part in Fig. 2) where nuclear HVAC system and heat exchangers for cooling water etc. are installed. The PPS manages the entering/leaving of the radiation controlled area according to the classification of the access mode described in Tab. 1.

Table 1: Access Mode of PPS

Access Mode	Status
No Access	"No Access" means a prohibition against all access to the accelerator vault. * On beam operation * While radiation dose is too high to enter the radiation area after beam operation.
Controlled Access	"Controlled Access" means accessible to the accelerator vault and radiation hot area, when the supervisor gives permission. * During preparation for beam operation. * Interval of beam operation. * Maintenance at the radiation hot area
Authorized Access	"Authorized Access" means possible to access to the accelerator vault and radiation hot area, without permission. * Under suspension of accelerator subsystems * During maintenance works

During beam operation, "No Access" mode means prohibition against all access to the accelerator vault. Only when no person in the accelerator vault is confirmed and "No Access" mode is selected, the beam operation is permitted by the PPS. When the access door is opened or the lock is released, the permission for beam operation is immediately cancelled by the PPS.

In the "Controlled Access" mode, the access to radiation hot area for maintenance etc. is possible on beam operation when the permission is given by supervisor, but the access to accelerator vault is prohibited. During preparation for beam operation, access to both areas is possible when the permission is given by supervisor.

To let two status modes of "No Access" and "Controlled Access" on beam operation distinguish surely, two personal key boxes are prepared. One is used for entering to the accelerator vault, and the other is used for entering to radiation hot area, so that it is possible to enter to hot radiation area during beam operation under the management of PPS. The key for the accelerator vault is impossible to pull out during beam operation by the PPS.

For "Authorized Access" mode, the access to both controlled area for maintenance etc. is possible. In this mode, two kind of personal keys are also needed to access the both area, because the access mode of PPS is never changed to "Controlled Access" or "No Access" before all personal keys are returned to key boxes.

EQUIPMENTS LAYOUT

PPS Racks

The PPS consists of four racks; Uninterruptible Power Supply (UPS) rack, PPS control rack (PLC-A), Cable connection rack, and PPS control rack (PLC-B), arranged from left to right in Fig. 2 (1). These racks are installed in RF & Power Supply Area.

PLCs are installed in the PPS control racks (A and B) with same composition, and the PPS is full duplication including signal paths. The UPS has capacity of 5kVA and ability to supply the PPS equipments without electric power interruption during 10 minutes until startup by a backup generator.

PPS Console

The PPS console is used for monitoring and operating against personnel entering/leaving for controlled area from the control room. The PPS console consists of PPS control panel, an intercom, LCD monitors etc., as shown in Fig. 2 (2). In the PPS console, it is possible to survey the situation in the accelerator vault and shipping bay area by monitoring cameras and intercoms. In addition, the console is also used for monitoring the important points for radiation boundary, for example "Door monitored by PPS" in Fig. 2.

The PPS control panel provides the follow switches; Access Mode switches, permission switches for personal key boxes and door control boxes, an emergency stop switch, etc. By using these switches, the supervisor can manage the access doors for personnel entering/leaving for controlled area.

Personal Key Boxes

Personal keys are used for unlocking access door in the controlled area. When workers enter into the controlled areas, each worker has to take a personal key. Unless all keys are returned to the key box, the beam operation is not permitted.

Two personal key boxes are installed in the access room for entering the accelerator vault (yellow key panel) and for entering the radiation hot area (blue key panel) as shown in Fig. 2 (3).

The workers cannot take the keys on the yellow key panel during the beam operation, while they can take them on the blue key panel, when the supervisor gives permission by the PPS to enter to the radiation hot area.

Door Control Boxes

For the access doors (indicated red circles in Fig. 2) to enter the controlled areas, access door control boxes are installed near the door. When permission to release the access door lock is given by the PPS console, the door lock is released and the access to controlled area is possible.

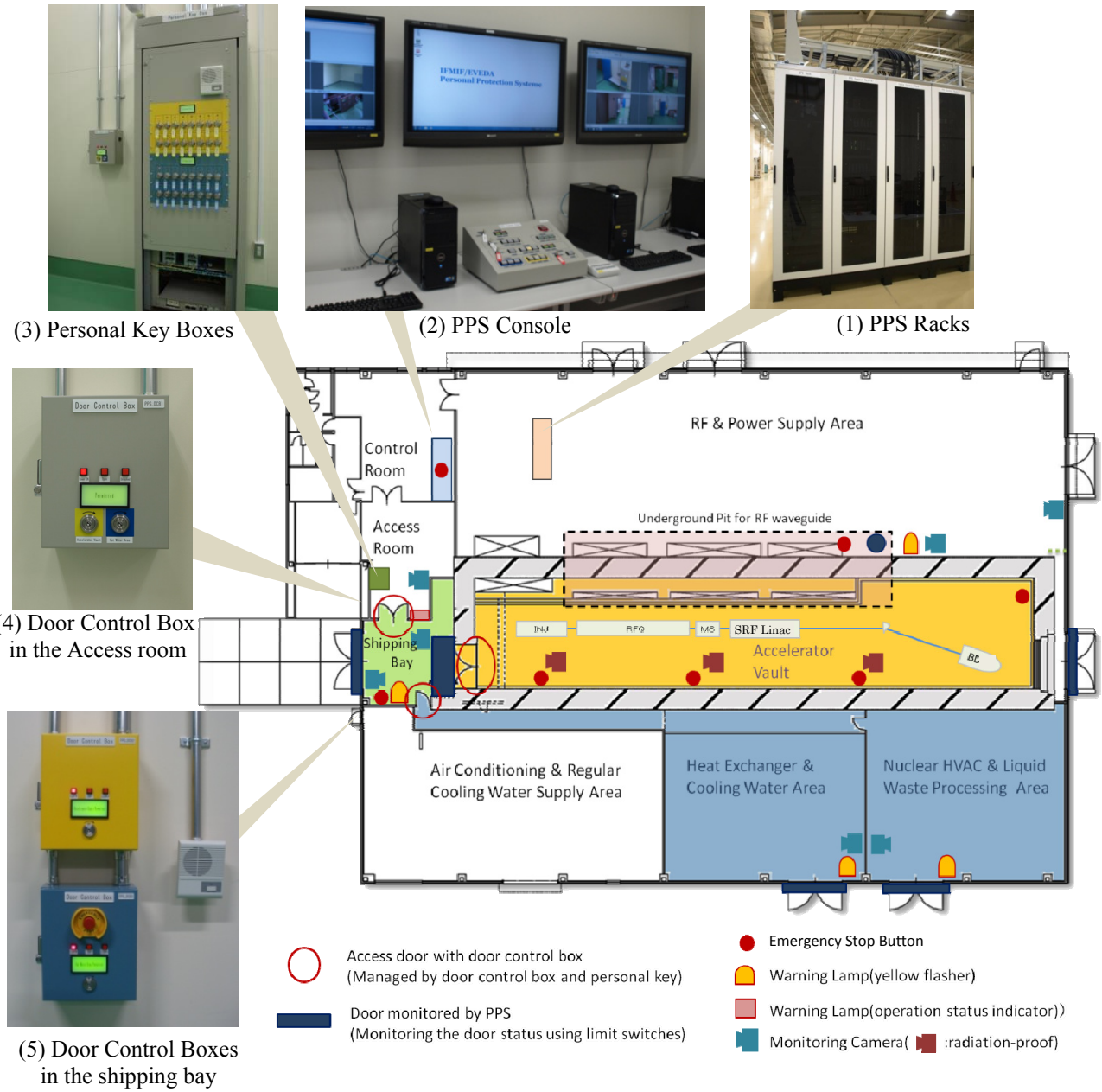


Figure 2: Radiation controlled area and Layout of main equipment for the PPS.

As indicated in Fig. 2, the workers have to access to the accelerator vault or the radiation hot areas from access room through the shipping bay. In the access room, the door control box is set, and the door lock can be released by yellow or blue keyhole as indicated in Fig. 2 (4). In the shipping bay, two access door control boxes are set for each access door (Fig. 2 (5)). After the supervisor can confirm the number of workers in the shipping bay by the monitoring cameras, a permission signal to the PPS is given by the supervisor, each door lock can be released by each personal key and person can enter to each areas.

(1) Emergency Stop Button (2) Monitoring Camera in the radiation proof box

Figure 3: Emergency stop button & Monitoring camera.

Emergency Stop Buttons & Monitoring Cameras

Four emergency stop buttons (Fig. 3 (1)) are installed in the accelerator vault in front of the accelerator subsystems; Injector, RFQ, SRF Linac, and Beam Dump.

When an abnormal or an emergency phenomenon happens, it is possible to stop or cancel for the beam operation by pushing the nearest button. The emergency stop button has a built-in alarm. The alarm rumbles loudly during the emergency stop button is pushed and also does during 5 minutes before starting beam operation to alert.

To monitor the important area for radiation safety at the control room, the monitoring cameras are installed in of the accelerator building. In the accelerator vault, 3 cameras are installed in the radiation proof box with lead shielding (Fig. 3 (2)). The 3 web cameras installed in the shipping bay and the access room make it possible to monitor views of the personnel entering/leaving for the controlled area. Some web cameras to monitor the radiation boundary are also installed in the RF & Power Supply Area and the hot radiation area.

SUMMARY

The PPS for the IFMIF/EVEDA prototype accelerator has been designed. In order to manage workers for maintenance, three status modes of "No access", "Controlled access" and "Authorized access" are applied for the accelerator vault area and the radiation hot area. For each status modes, management of workers for the entering/leaving is carried out by the personal keys, the monitoring system.

In the present status, sequence logics and programmed ladder on PLCs are applied, and the PPS has already performed the equivalent radiation safety management.

According to the interface signals between PPS and the accelerator subsystems, the responsible officers both JAEA and European Institutes are discussing details. Each interface signal will be performed at the linkage test with the accelerator subsystems in Europe, before they will be delivered to Rokkasho site [3].

REFERENCES

[1] T. Kojima et al., "Design policy of the Personnel Protection System for the IFMIF/EVEDA Accelerator", Proceedings of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Japan, August 2009.

[2] H. Takahashi et al., "Overview of the Control System for the IFMIF/EVEDA Accelerator", Proceedings of the 6th Annual Meeting of Particle Accelerator Society of Japan, Tokai, Japan, August 2009.

[3] T. Kojima et al., "Linkage test of Control system and Injector for IFMIF/EVEDA Accelerator", Proceedings of the 7th Annual Meeting of Particle Accelerator Society of Japan, Himeji, Japan, August 2010.

ASSESSMENT AND TESTING OF INDUSTRIAL DEVICES ROBUSTNESS AGAINST CYBER SECURITY ATTACKS

F. Tilaro, B. Copy, CERN, Geneva, Switzerland

Abstract

CERN (European Organization for Nuclear Research),like any organization, needs to achieve the conflicting objectives of connecting its operational network to Internet while at the same time keeping its industrial control systems secure from external and internal cyber attacks. With this in mind, the ISA-99 [1] international cyber security standard has been adopted at CERN as a reference model to define a set of guidelines and security robustness criteria applicable to any network device. Devices robustness represents a key link in the defense-in-depth concept as some attacks will inevitably penetrate security boundaries and thus require further protection measures. When assessing the cyber security robustness of devices we have singled out control system-relevant attack patterns derived from the well-known CAPEC [2] classification. Once a vulnerability is identified, it needs to be documented, prioritized and reproduced at will in a dedicated test environment for debugging purposes. CERN - in collaboration with SIEMENS - has designed and implemented a dedicated working environment, the Test-bench for Robustness of Industrial Equipments [3] ("TRoIE"). Such tests attempt to detect possible anomalies by exploiting corrupt communication channels and manipulating the normal behavior of the communication protocols, in the same way as a cyber attacker would proceed. This document provides an inventory of security guidelines [4] relevant to the CERN industrial environment and describes how we have automated the collection and classification of identified vulnerabilities into a test-bench.

INTRODUCTION

The Internet has become essential for any organization or company that would like to conduct any sort of business. On one hand, in Industry this results in an improvement of the communication from the fabric floor network to the business level one; but on the other hand, the Internet has brought a lot of security problems which cannot be ignored because of the resulting damages and disasters. Thus, it is necessary to identify methods and procedures for achieving the dual mission of both exploiting the advantages provided by the Internet and at the same time keeping their industrial systems secure from external and internal attacks.

As we know, the number of hostile applications, worms and viruses is continuously increasing and hackers are more and more interested at exploiting common industrial control systems vulnerabilities [5]. As the well-known industrial control security expert Joe Weiss testified to the US Congress, in 2010 more than 180 security incidents were reported; it is worth to mention the Deepwater Horizon (BP Mexican Gulf oil spill) counting 11 casualties or the San Bruno gas pipeline explosion, with 8 casualties.

It is evident that industrial security cannot rely only on IT security techniques and properly configured network architectures (like firewall, network segmentation, antivirus software, access management, etc…) which, however, can provide only a partial protection for the entire system; this is easily explained if we consider the fundamental differences in the deployed hardware but also in the concepts of availability and manageability between IT and Industrial Systems. On the contrary, new methodologies aimed at securing industrial systems must be integrated into IT security standard practises.

This paper mainly focuses on testing industrial Ethernet-connection-based devices without going into details on network security configurations.

COMMUNICATION ROBUSTNESS

Any communication between two or more hosts is regulated by a protocol whose specification defines the formats, syntax and semantic rules for exchanging messages – called Protocol Data Units (PDUs) [6] - over a network. Protocol specifications are typically written in natural - not deterministic - language and sometimes cannot state how to handle all the possible faulty inputs; so many decisions are left to the implementer. Hence it is clear that distinct implementations of the same protocol handle PDUs in different ways.

Due to this heterogeneity, each implementation could present many issues related to services availability, performance and even security: once a hacker has figured out some malformed PDUs or sequences of them which are not properly handled, the entire system is under risk [4]. Ideally all these defects affecting the protocol implementations should be detected and fixed by the manufacturers, but they are sometimes actually detected and reported by external communities. This explains the need of performing automated testing of protocol implementations, especially for critical systems.

In this paper we present a methodology for automated testing of protocol implementation robustness, which must be seen as the ability of the system to handle exceptionally malformed PDUs and stressful network conditions, while maintaining the normal operational behaviour. This evaluation proves a greater ability of analyzed networked systems to survive in the face of

malformed input due to possible involuntary mistakes or explicit attack attempts.

As explained in the following section of the document, the enumeration of all possible faulty PDUs for each protocol is exponential in the number of protocol fields; so it is necessary to devise a strategy to reduce the number of possible malformed PDUs to generate, while still detecting security issues accurately. This can be achieved by exploiting the knowledge of communication experts to distinguish the test cases which could be really interesting, from the ones that are redundant and maybe even duplicated. The approach we are proposing is used to generate individual malformed PDUs or sequences of them in a systematic manner according to a specific grammar definition. Grammars define a set of syntactic and semantic rules to cover a specific domain of tests (generally related to a specific communicational protocol header); if the protocol implementation cannot handle invalid packets correctly, anomalous behaviour may occur and needs to be detected.

The approach, we are proposing, must be generic enough to be applied to all protocols even to industrial ones which exhibit really specific properties and features.

ANALYZED TOOLS AND SECURITY STANDARDS

In the recent years the increasing number of cyber security related incidents affecting industrial control systems has made vendors of critical infrastructure to pay more attention to security issues. Unfortunately, there are no complete and comprehensive standards whose specifications can be followed to protect any critical system; nonetheless several initiatives have been started with the objective of improving the security level and the robustness of industrial systems.

The ISA Secure Program [7], on the basis of ISA 99 Standards specifications, has produced a certification program with three levels of recognition for a device security assurance.

Among the commercial products - that we have already tested and used in our initial testing phase - Wurldtech Achilles [8] must be mentioned; it is an all-in-one vulnerability scanner specifically designed for suppliers of industrial process automation, control and safety systems such as: Programmable Logic Controllers, Supervisory Control and Data Acquisition (SCADA) Devices, Distributed Control Systems and Critical Systems. Wurldtech has also developed a proprietary certification with the purpose of evaluating and assuring the robustness and resilience of industrial products. However, to overcome the Achilles platform's proprietary aspects and limitations in terms of supported network protocols and attack techniques customization support, we have designed and implemented the TRoIE test-bench [4]. Moreover Wurldtech Inc. is also providing a valuable contribution to the preparation of industrial cyber security standards, such as ISA99.

In the wide range of open-source security frameworks and tools we have integrated into our test-bench the Open Vulnerability Assessment System [9] (OpenVAS); born as a branch of the popular Nessus2, OpenVAS provides a flexible environment for the development and deployment of new vulnerability scanning techniques. So far this framework does not offer a real wide range of tests developed specifically for industrial and process control environments. This field has been attracting more and more interest in the last few years: for instance, the Nessus vulnerability scanner has recently introduced an internal package finalized to industrial protocols and systems tests.

ENHANCED PROTOCOL FUZZING TESTS WITH GRAMMAR DEFINITION

When assessing the cyber security robustness of devices, *attack patterns* [10] can be used to categorize the discovered vulnerabilities. An attack pattern is expressed as "a series of repeatable steps simulating an attack against a system".

Such classification is useful to identify the cause of vulnerability and the potentially related well-known solution. In our experience we focused on Fuzzing and Grammar tests because of their effectiveness at probing devices for security vulnerabilities.

Protocol Fuzzing is a very effective testing technique generally deployed to generate valid and invalid packets with "randomized" header field values; its main purpose is analyzing the behaviour of a specific protocol implementation or functions of the protocol stack by injecting unexpectedly malformed input parameters values.

Fuzzing, according to the first, narrower definition, might be characterized as a blind fishing expedition that hopes to uncover completely unsuspected problems in the software under test. However it must be also admitted that for many interfaces, the idea of simply injecting random bits works poorly generally because of the wide domain of test. For example, injecting a web interface with randomly generated strings will have the only effect to detect only invalid URLs: they will be mainly rejected suddenly, perhaps by a simple parser - acting a sort of protection - checking for valid URLs. This is why completely random fuzzing is a comparatively ineffective way to uncover problems in a generic system. On the contrary Fuzzing acquires more efficiency when it is combined with "intelligent" techniques. Microsoft refers to this as "smart fuzzing" [11]: it is not a random testing anymore, but the generation of the tests is led by the target specifications; hence an initial knowledge of the system under test is required. In our case the fuzzing system is fed with some grammar rules which specify the part of the protocol headers to fuzz and the strategy we want to follow for the generation of the injected packets. One of the most important benefits coming from this kind of testing strategy is the ability to systematically and

predictably explore the input space, instead of having to rely on randomly generated noise.

Our final strategy is a mix of fuzzing and syntax testing: syntax is used to translate the knowledge of security experts into rules, which determine the packets generation and injection. In any case, the creation of effective syntax tests requires a deep understanding of the multiple protocols under test and their possibly stateful interconnections.

A PROTOCOL FUZZING SYSTEM IMPLEMENTATION

In the initial phase of the project, we analyzed a wide range of available specialized fuzzing utilities. Some of them exhaustively iterate through a designated protocol, whose specification are known in advance; so they could be used to stress test a variety of applications that support that protocol. We then opted for other generic fuzzers capable of fuzzing arbitrary protocols and file formats by performing simple, in principle non-protocol-aware, data mutations such as bit flipping or byte transposing. Although these fuzzers are effective against a wide range of common applications, we often have a need for more customization and thorough fuzzing for proprietary and previously untested protocols (we remind that our main targets are industrial control devices with proprietary specifications). This is where fuzzing frameworks become extremely useful. Among them we explored three open source fuzzing frameworks: SPIKE, Sulley Fuzzing Framework and Peach. Eventually the last framework has been selected for its flexibility and simplicity at developing specific customized protocols fuzzing tests. Nonetheless Peach provides some utility to convert Wireshark captured network traffic file into its protocol model format. It also includes several modules for target faults detections; but unfortunately they cannot be used in our scenarios where a custom monitoring system has been developed to observe the behaviour of the industrial devices under tests.

Peach Fuzzing framework [12] is an open source cross-platform fuzzing framework written in Python. It provides with a well designed software components, like mutators, transformers, protocols definitions, publishers, groups, etc... These components can be extended and chained together to simplify the generation of customized complex data types. As underlined before, Peach offers a very high object oriented abstraction level through the definition of pure python classes: it allows a tester to focus on the individual subcomponents of a given protocol, later tying them together to create a complete fuzzer scheme. As a result this approach generously promotes the code reuse for the development of new fuzzers. We have customized this framework by defining:

- new publishers tailored at injecting the generated data into the specific protocol formats we want to test;
- new mutators responsible for the generation of data ranging values and types;
- new transformers to encode the data values in a proper way;
- new mutation strategies which establish the algorithm to follow at combining the protocol fields values;
- new agents and monitors to integrate the fuzzing system with the external monitoring one: in case of target failure detection the fuzzing test saves its current state to restart it later from the last point.

Once the framework has been customized through the implementation of the previously described components, we have started the creation of so called “PeachPit” files: they are XML files containing all of the information necessary to run a fuzzing test:

- the data model defines the structure, information type, and relationships in the data to be fuzzed;
- the state model: It additionally allows for the configuration of a fuzzing run including selecting a data transport (Publisher), logging interface, etc.
- Generic configuration to specify the publishers, agents and monitors to use, including their specific initial parameters values.

In line with the ISA-99 security standards specifications we have defined an independent testing and certification system for industrial control devices. The PeachPit file definition is not totally arbitrary, but aims at fulfilling the ISCI Communication Robustness Testing (CRT) requirements. In practises we have implemented within the Peach fuzzing framework a list of security tests classified by protocol and scope of test. However it must be said that this certification does not cover the industrial protocols yet; so, to overcome this limitation, we have defined our own tests by applying the same security concepts and guidelines used for the open protocols.

MONITORING

In any testing procedure it is essential for the tester to be able to determine if the behaviour of the system under test is symptomatic of vulnerability or anomalies either in the software or hardware.

This examination can be harder in security testing than in traditional functional testing, because the tester is not necessarily comparing actual program behaviour to expectations derived from specifications. Rather, the tester is often looking for unspecified symptoms indicating the presence of unsuspected vulnerabilities. Furthermore monitoring must be integrated into several automated procedures, which can be used to evaluate test outcomes and identify only the anomalous ones: this is especially true during high-volume test activities like fuzzing. Unfortunately there are no third-party test automation tools able to monitor generic industrial devices’ outputs and internal behaviour; so we have been forced to design and implement our own monitoring strategy. At least the “essential services” - following the ISCI CRT [6] naming convention - must be observed

during the testing activities. In our specific case the monitoring system should check and track any of the following effects: extension of the current PLCs scan cycle, excessive use of memory, CPU overuse, delays in the physical outputs and consistent delays in the periodic communication.

Once an anomaly has been detected, it becomes relevant to identify its cause, the specific packet or sequence of packets. Because of its complexity it is not generally easy to achieve this task: one anomaly could be the result of different environment variables, and the device's behaviour could not be affected by the single factors, but only by a specific combination of them.

ACHIEVEMENTS

The described strategy has already proven to be effective at detecting device robustness issues. Thanks to the performed testing analysis, it was possible to detect critical anomalies in the devices' software protocol stack implementations. These research findings have been directly reported to their proper industrial vendors in order to be patched and incorporated in subsequent firmware releases. These initial encouraging results have motivated the team to continue following and expanding this approach for the future of the collaboration between CERN and the automation industry.

CONCLUSIONS AND FUTURE WORK

The entire article is based on the assumption that any network protocol implementation is susceptible to accidental or malicious corrupted communication. For this reason, it is essential to perform robustness testing of these implementations for critical industrial systems. Our approach consists of analyzing protocol implementations by injecting malformed PDUs to corrupt the normal behaviour of the system. As a PDU typically has many fields, the number of possible syntactically faulty PDUs grows exponentially with the number of fields. In this document, we proposed a strategy to explore this huge test domain using a hybrid approach of fuzzing and syntax techniques, specifically developed to evaluate industrial device communication robustness. So far, not all the tests can be integrated into automatic tools, human analysis and management is necessary to discover and investigate specific possible failures.

Moreover it should be remembered that security analysis must be seen as a dynamic process which should be adapted according to new requirements, constraints and technological changes. So the testing techniques and methodologies defined in this document should be adapted and modified to fit incoming features and evaluate new functionality.

In the future, we will extend the scope of our analysis to the industrial supervision layer: the targets of our tests will not only be the individual devices but also SCADA systems in order to estimate the potential impact of malicious PDUs within the entire industrial network architecture.

REFERENCES

[1] ISA-99: Manufacturing and Control Systems Security, http://www.isa.org
[2] CAPEC (Common Attack Pattern Enumeration and Classification), http://capec.mitre.org
[3] F. Tilaro, "Test-bench for Robustness...", CERN, 2009
[4] B. Copy, F. Tilaro "Standards based measurable security for embedded devices", ICALEPCS 2009
[5] Stuxnet, http://www.langner.com/en/2011/08/04/stuxnet-and-beresford/
[6] A. Tanenbaum "Computer Networks", Prentice Hall 2000
[7] ISA Secure Program, http://www.isasecure.org/
[8] Achilles Satellite Security & Robustness Testing Platform, http://www.wurldtech.com/cyber-security-products/achilles-satellite/default.aspx
[9] Open Vulnerability Assessment System (OpenVAS), http://www.openvas.org/
[10] F. Tilaro, B. Copy "Guidelines for evaluating PLC security", CERN, 2010
[11] Howard, Michael & Lipner, Steve. The Security Development Lifecycle. Redmond, WA: Microsoft Press, 2006
[12] Peach Fuzzing Platform, http://peachfuzzer.com/

CERN SAFETY SYSTEM MONITORING - SSM

T. Hakulinen, P. Ninin, F. Valentini, CERN, Geneva, Switzerland
J. Gonzalez, C. Salatko-Petryszcze, Assystem, France

Abstract

CERN SSM (Safety System Monitoring) [1] is a system for monitoring state-of-health of the various access and safety systems of the CERN site and accelerator infrastructure. The emphasis of SSM is on the needs of maintenance and system operation with the aim of providing an independent and reliable verification path of the basic operational parameters of each system. Included are all network-connected devices, such as PLCs, servers, panel displays, operator posts, etc. The basic monitoring engine of SSM is a freely available system-monitoring framework Zabbix [2], on top of which a simplified traffic-light-type web-interface has been built. The web-interface of SSM is designed to be ultra-light to facilitate access from handheld devices over slow connections. The underlying Zabbix system offers history and notification mechanisms typical of advanced monitoring systems.

INTRODUCTION

This paper presents, Safety System Monitoring (SSM), a framework for monitoring the computing infrastructure of CERN safety-related systems, such as the LHC and PS access and safety systems. While other monitoring tools exist at CERN, none of them is able to easily monitor control systems as heterogeneous as the access and safety systems.

The main purpose of SSM is to present the access and safety maintenance teams an accurate overall picture of the functioning of the various parts of the monitored systems. While the monitoring system is primarily aimed at the maintenance teams, some elements are still available to access operators and safety personnel.

MOTIVATION

In order to ensure the best possible service to their clients, access and safety maintenance teams need to know what is going on with their systems before their clients tell them. The goal of SSM is to provide the teams a straightforward remotely accessible user interface, which can help to anticipate major problems by early detection of failures. The following access and safety systems are the target of the SSM system:

- **LACS** (LHC Access Control System) – who enters the LHC and when?
- **LASS** (LHC Access Safety System) – is it safe for beam or access?
- **PACS** (PS Access Control System) – idem for the PS complex.
- **PASS** (PS Access Safety System) – idem.
- **SPS PSS** – integrated personnel safety system for the SPS complex.
- **SUSI** (Surveillance des Sites) – who enters CERN sites and areas other than the accelerators.
- **CSAM** (CERN Safety Alarm Monitoring) – alarms for the fire brigade.
- **Sniffer** – gas detection and alarm.
- **SIP** (Site Information Panels) – display relevant info at access points.
- **SSA** (Safety System Atlas) – Access and personnel safety system for the ATLAS detector.

The systems themselves have normally some internal tools giving at least info on the operational status and internal state of health, as well as synoptic displays for operators and maintenance. However, there is little coherence between the different systems due to their often different approaches and levels of abstraction. The native interfaces also tend to be difficult to use requiring expert knowledge, their information cannot always be trusted due to too many layers and black-box implementation of those systems, and not everything is necessarily monitored – particularly in systems composed of many diverse components. CERN access and safety system typically consist of:

- Servers (Windows / Linux).
- Operator posts (PCs at control rooms / access service).
- Panel-PCs (local displays / information panels).
- PLCs / UTLs (local special purpose control units).
- Various other local control units.
- Video cameras / recorders.
- Biometry units (iris-scan).
- Interphones (at access points and operator rooms).
- Card readers.
- Key distributor units.
- Databases / web-servers.

These various devices come from many different vendors, and they are nowadays mostly directly network connected. Access systems reside mainly in the CERN Technical Network (TN) but some equipment reside also in the General Purpose Network (GPN) and the most important systems, in particular safety systems, have their own private networks.

It is to manage this complexity in a unified manner that the SSM system was conceived: The goal was to build an integrated system for monitoring all safety and access systems managed by the CERN access and safety teams in a coherent and reliable manner. The following basic requirements for the SSM system were identified:

- SSM would need to target directly all network-connected devices and subsystems of the systems being monitored.
- SSM needed to be easily accessible from individual PCs inside and outside of CERN.

- The adopted approach was a simple system of traffic-light style indicators (green/yellow/red) to give a quick indication of problems without going into too much detail.
- The SSM interface had to be well adapted to fixed information panel displays.
- SSM had to be easily usable with handheld devices.
- The underlying monitoring engine was to be kept separate from the visualization interface.
- It should be possible to distribute information to other systems, such as the CERN Technical Infrastructure Monitoring (TIM) [3], when this information is not available elsewhere.

A small on-paper comparison of the monitoring systems currently in use at CERN was made, and monitoring system Zabbix came out as the tool of choice [4]. The main reasons were the openness and easy configurability of Zabbix, its out-of-the-box support for both Windows and Linux/Unix systems, SNMP and IPMI support, Oracle support, it being free of charge, and the fact that it was already known by the access and safety teams thereby making it easily acceptable.

A development project was initiated together with Assystem to develop the basic SSM application infrastructure and user interface.

DESIGN PRINCIPLES

The general design of the SSM system is based on the following basic principles:

- **Simplicity**: Use well-defined interfaces with clear functional separation. Use existing systems and CERN standard services whenever possible (example: Oracle, web-services, authentication).
- **Reliability**: Put in place self-diagnostic checks to tell if the displayed information is trustworthy.
- **Independency**: Look at the system to be monitored from the outside and avoid using information produced by that system. Go to the source whenever possible (example: access PLCs directly).
- **Maintainability**: Keep scripts and database structure simple and easy to understand with up to date documentation.
- **Accessibility**: Must work with all major web-browsers and handheld devices from anywhere.
- **Confidentiality**: Access is to be limited to a well-defined group and a login with CERN password required.

A division of responsibilities between system layers is based on a functional separation:

- **Collect** function: The underlying monitoring engine Zabbix carries out the actual monitoring tasks (management of local agents, connections, item logging, events, notification).
- **Synthesize** function: The integration and synthesis layer consists of a separate "scratch" Oracle database, which has access to Zabbix database tables. The database imports parts of the internal group-machine-item-trigger structure from Zabbix. All the synthesis rules for properly combining the information are defined as Oracle procedures.
- **Visualize** function: The visualization layer consists of the SSM web site [1], which accesses the synthesis database using a PHP-Oracle interface. The site is designed to be as light and simple as possible and to support both interactive and static displays.

SSM visualization layer presents information to users based on a 3-tier approach, where the roles of the different tiers are well defined:

General System Availability Tier

The general system availability tier is a generic web-based top tier presenting general status-information in the form of a synthesis of the states of various subsystems. This interface is designed to be extremely simple and light to facilitate viewing with any web browser or hand-held device (see Fig. 1).

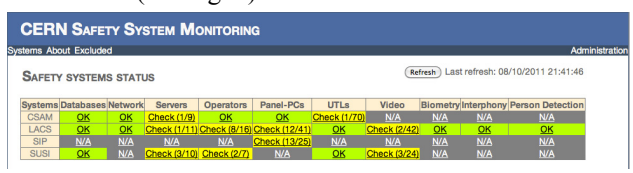


Figure 1: SSM main system synthesis view. Every monitored system is presented on a line of its own, with different subsystem categories shown as columns. Green **OK** means that everything is fine with every equipment of the subsystem, yellow **Check** means that the system continues to function but in a reduced state (non-critical failure of some equipment, with the number of failing equipment given in parenthesis), red **Down** would signify a critical failure rendering the subsystem non-operational, and grey **N/A** means that this item is not applicable to the system in question.

Subsystem Availability and Diagnostic Tier

The subsystem availability and diagnostic tier is a more detailed web-based 2nd tier allows access to a selected set of availability and diagnostic information of the various subsystems. This is simply an extension of the top tier pages allowing one to dig more deeply into the system. Similarly to the top tier, viewing of the pages is possible with the same kinds of devices (see Fig. 2).

Low-Level Diagnostic Tier

The 3rd tier comprises access to native diagnostic and configuration tools of the various subsystems. In most cases this means simply a direct link to a web-based interface of the underlying monitoring tools like Zabbix (see Fig. 3).

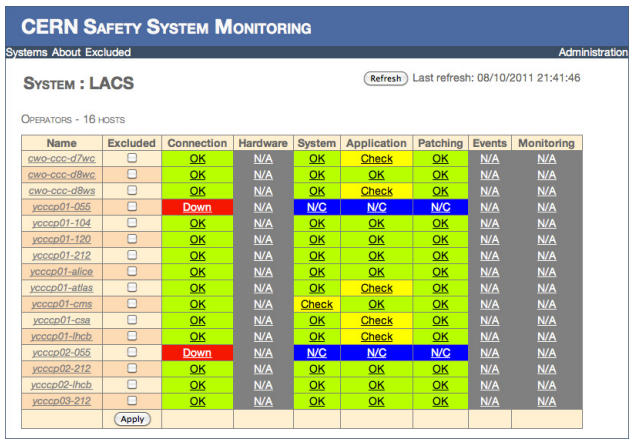


Figure 2: SSM subsystem view. All the devices of a subsystem are shown on a line of their own. The significance of the colours is the same as in the main view with addition of blue **N/C** meaning that the equipment is not communicating the information in question. The special item **Excluded** gives the possibility to mark some equipment to be excluded from the synthesis calculation (thereby not being visible in the main synthesis view) to prevent equipment undergoing maintenance or some known failures from polluting the synthesis view.

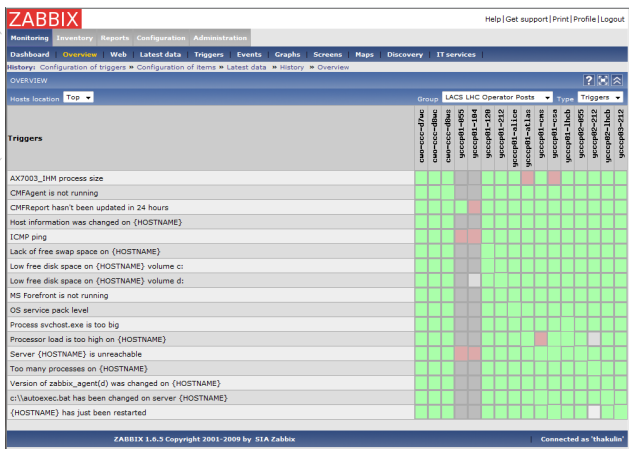


Figure 3: The native Zabbix interface. This interface permits a full access to all the features of the Zabbix tool including trend graphs, host configuration information, and email and SMS notifications. This interface is only accessible from the CERN network.

Target Audience

The expected audience of the monitoring system can be roughly divided in three categories depending on the roles and needs of each. Access to the system can be tailored to each case separately:

- **Access and safety maintenance and operation teams**. By virtue of their functions and being the main target of the monitoring system, the maintenance and operation teams have full access to all of the information and functionalities of the monitoring system.
- **Access operators, safety personnel, operational personnel, and registration service**. As the main "clients" of the access and safety systems, the operational personnel can benefit from certain elements of the monitoring system, but will probably not need access to the underlying tools.
- **CERN users and general public**. There is no interest to expose an internal monitoring system to the general public. However, some information provided by the system might become available via indirect routes, such as the information panels visible in publicly accessible areas.

Since access to the monitoring system is generally restricted to the members of the access and safety system teams and operators, an access control scheme has been implemented. This was achieved by controlling access to the web pages. There are two complementary authorization mechanisms:

- The web-server checks the ip-address of the machine where the request is coming from and if it is on the list of machines authorized to access the pages directly, access is given. These machines would include the machines running the safety information panels in stand-alone mode and possibly certain operator posts, access control configuration workstations, and dedicated terminal servers under direct control of the maintenance team or operation.
- Otherwise, a personal CERN login is requested. If the person is on the list of authorized persons, access is given. To use any configuration or diagnostic utilities with other powers besides a simple display of information, personal login is always required.

NETWORK ARCHITECTURE

The overall network architecture of SSM is presented in Figure 4. It consists of the following components:

- **Main Zabbix server**: The main monitoring engine resides in the Technical Network.
- **Proxy Zabbix server(s)**: A proxy server is a slave monitoring server for a private network able to act as an information gateway for the monitoring system between the TN and the private network segment.
- **Zabbix database:** The main monitoring data repository residing on the CERN central Oracle servers.
- **SSM database:** A "scratch" database with access to the Zabbix database and containing the synthesis rules and the data visualization logic of the SSM web interface.
- **SSMTIM database:** A second scratch database accessing the Zabbix database to distribute selected data to the TIM framework.
- **SSM web site:** A dynamic web site based on PHP scripts, which provide access to the SSM database for visualization. The web scripts contain no other logic except what is required to format and visualize the information.

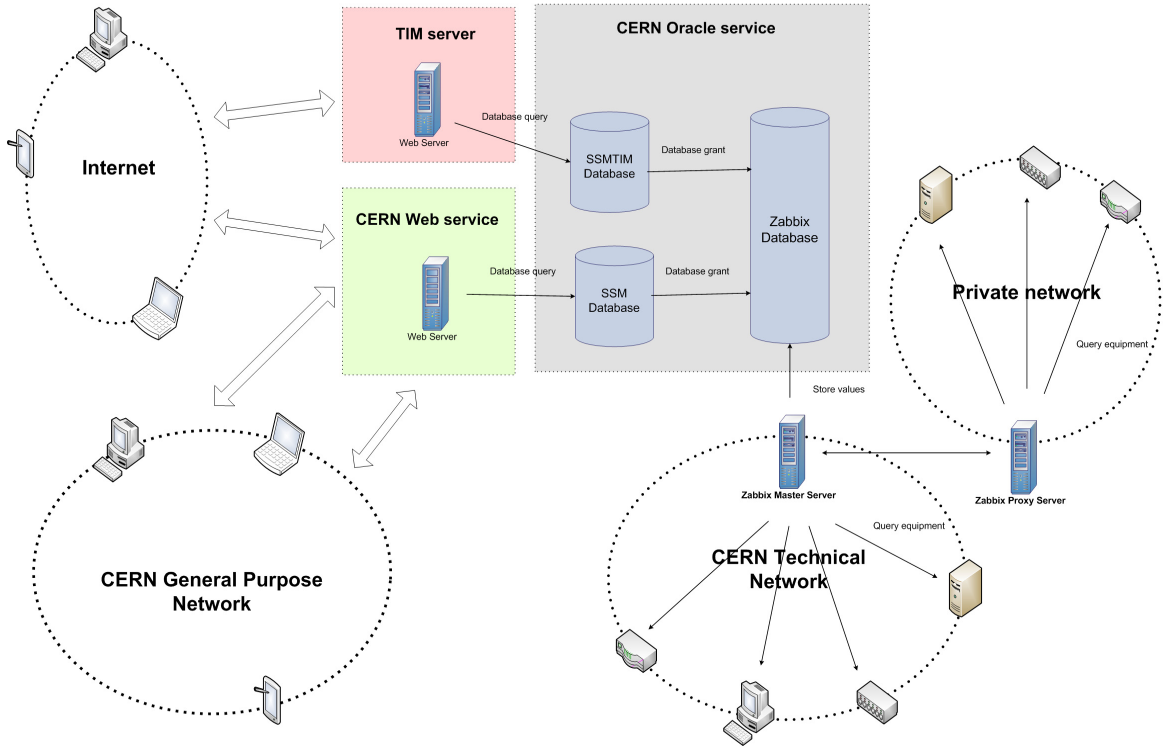


Figure 4: The overall SSM network architecture.

CURRENT STATUS

After a pilot phase of several months, SSM is in production at CERN since summer 2011. The systems integrated into the SSM are: LACS, CSAM, SUSI, and SIP. Work continues to integrate the remaining systems, SPS PSS, Sniffer, and SSA. The PS complex access and safety systems (PACS/PASS) will be integrated into the SSM framework once the renovation of those systems is final during the CERN accelerator shutdown of 2013-2014. Below are some current statistics of the SSM system today:

- 700 network connected equipment of the various systems are now monitored:
  - LACS: servers, operator posts, panel-PCs UTLs, interphones, biometry, video recorders, person detection units, network switches.
  - CSAM: servers, operator posts, panel-PCs, PLCs, network switches.
  - SUSI: servers, operator posts, UTLs, video recorders, video cameras, network card readers.
  - SIP: info screens.
  - Some small safety-related systems by other CERN teams monitored as a free service to those teams.
- 5900 monitoring items (180 new values / second).
- The full history of all values is kept for 30 days and trend data for 6 months.
- The access and safety team members have personal SSM and Zabbix user accounts to allow personal notifications via email / SMS. The authentication is via standard CERN login.
- The system has turned out to be very robust: (almost) no glitches in over a year of server uptime.

CONCLUSIONS

The CERN Safety System Monitoring (SSM) system has been designed to offer a robust and easily accessible tool for the monitoring of large scale heterogeneous systems made of numerous industrial components. It is currently under deployment on most of the CERN safety-related systems, but its open, pragmatic, and ergonomic approach based on a 3-tier user interface could be used also for the monitoring of industrial or accelerator related systems.

REFERENCES

[1] SSM web interface (CERN login required); http://cern.ch/ssm
[2] Zabbix; http://www.zabbix.com
[3] TIM (only inside the CERN network); http://timweb
[4] SSM Technical Specification; https://edms.cern.ch/document/1015740

VIRTUALIZATION IN CONTROL SYSTEM ENVIRONMENT

L.R.Shen, D.K.Liu, T.M.Wan, SSRF Shanghai, China 201800

Abstract

In large scale distribute control system, there are lots of common service composed an environment for the entire control system, such as the server system for the common software base library, application server, archive server and so on. This paper gave a description of a virtualization realization for control system environment include the virtualization for server, storage, network system and application for the control system. With a virtualization instance of the epics based control system environment that built by the VMware vSphere v4 [1], we tested the whole functionality of this virtualization environment in the SSRF [2] control system, include the common server of the NFS, NIS, NTP, Boot and EPICS base and extension library tools, we also virtualization the application server such as the Archive, Alarm, EPICS gateway and all of the network based IOC. Specially, we test the high availability (HA) and VMotion for EPICS asynchronous IOC successful under the different VLAN configuration of the current SSRF control system network.

THE SSRF CONTROL SYSTEM ENVIRONMENT

A uniform environment which based on PC Linux system had been set up for the system development and operation in SSRF control system [3]. It composed with sets of server system that running Centos system and supplying the EPICS development and high level physics application environment. NIS and NFS server were used to manage user account and share resource. As part of runtime environment, the database server, boot server, Soft IOC server and EPICS application server such archive and alarm handler were installed at the server room.

Several versions (from v3.13.x to v3.14.8.2) of EPICS based with cross-compiler support were installed in the runtime environment and the base 3.14.8.2 is used currently. The target include the Linux-x86 and ppc-604_long architecture which is defined in EPICS.

Moreover in this environment, the EPICS extensions had been installed includes all the tools we needed like the EPICS tools like sequencer, ALH, EDM, MEDM, archive engine and so on. A standard directory structure of the entire environment was defined for all users in a centre NFS server, looks like the "/usr/local/epics/base-3.xx.xxx" for epics base, "/opt/matlab" for the MatLab.

During the operation in past years at SSRF, there were several times of the server system crash in control system environment and brought the accelerator shutdown accident. At the once of most serious it generated the shutdown more than 24 hours waiting for resume the server system normal at the once of most serious. Especially, there were many centralized IOC servers used for large amounts of asynchronous devices in SSRF control system. It will cause the beam lost frequently when these IOC servers down. A step further, the server crash will cause the data lost permanently and consume more time to reinstall the IOC server.

Based on these problem above, we think about the virtualization of the whole control system environment to reduce the failure rate and increase the MTBF of control system environment though the high availability and auto migration on fault brought by the virtualization.

VIRTUALIZATION OF SSRF CONTROL SYSTEM ENVIRONMENT

SSRF controls system environment was used for operation, management and maintenance of the whole SSRF facility. The server system achieved the application of control system and the user management. For the long time stability of server system and satisfy of the requirement of 7X24 hour running, we adopted the virtualization technique and tested in control system environment [4] Include the server system virtualization and the application virtualization.

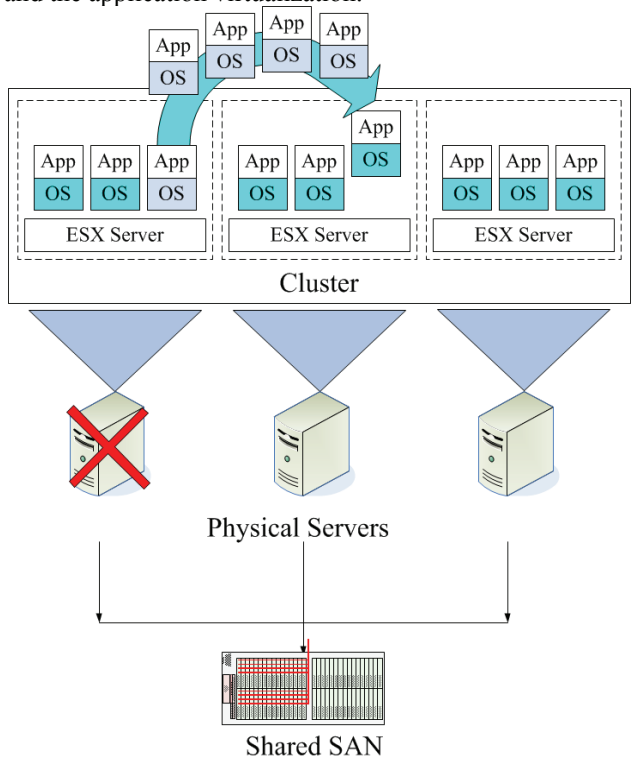


Figure 1: Structure of the virtual server system.

The SSRF virtual server system composed with 16 Dell PowerEdge 2950 PC rack server and a EMC CX-320 SAN storage system, every server used a HBA card connect to the centralized SAN storage system that realized the high availability virtual machine system with online migration capability (see Fig. 1). Another thing

need to noticed was that all the virtual machines operation system also must be installed on the SAN storage system to get the auto migration ability when the hardware fault. Cause we did not need the 100% availability. We did not use dual HBA link connection redundancy, just keep a SAN switch and some HBA card as the replacement part. The only risk is the replacement time of SAN switch when it on fault. The estimated time of change of SAN switch is about 10 minutes.

Hardware Resource Allocation Principle of Virtual Machine

According to the current status of the SSRF server system, the CPU and memory utilization is very low. For some typical system in SSRF, the memory used of archive engine is 484M, YUM and NFS is 1.2G, IOC Server is 338M. So we make the hardware resource allocation rule as below:

- Type1：1 CPU, 2G memory, 20G hard disk;
  Used for NIS, IOC server
- Type2: 2 CPU, 4G memory, hard disk based on requirement;
- Used for NFS, Archive server
- Type3: custom type, hardware resource was custom defined on demand.
- Used for some application such as Yum server, not need the high CPU and RAM utilization but may be need more hard disk.

For some special requirement for example the CA gateway, all the virtual machines used a single network adapter (virtual network adapter).

The Template of the Virtual Machine

According to the principle of hardware resource allocation above, we made several different kinds of virtual machine template so that we can create new virtual server on demond. The frequently used templates were the application server and IOC server that help the administrator make a new application or IOC server quickly and easily. Because all the EPICS base library, IOC software and environment were preinstall and configured in the template, so do not need any install and configured process, the administrator just need to click mouse several times then a new application or IOC server was created and operation immediately.

The template also carry with the storage and network information so we can add new application server has specially depended and dive into use immediately, such as add new EPICS gate way or the archive server when expand the system.

The VLAN setting for the IOC virtual machine

SSRF control system use VLAN to instead of device subnet. There are many asynchronous IOC server used for different sub system in different VLAN, such as the MPS IOC and the Vacuum IOC [5]. Before the virtualization, we need to run multiple IOC servers at one physical server, so we need to config VLAN setting for every physical server. After the virtualization we can run IOC on virtual server one by one. The VMWare HA characteristic need to move these IOCs move from one physical machine to another physical machine, so we need to configure virtual switch with multiple VLAN on same physical server. In the physical machine network configure, we add virtual switch port group one by one and assigned with different VLAN ID of subnet to the virtual switch (see Fig. 2).

Figure 2：VLAN configure in virtual switch.

Except the IOC server, the EPICS gateway server is also across two different VLAN and need to configure virtual network switch that suit for different VLAN so that the virtual machine of EPICS gateway can migration between the different physical machine.

The Implement of the Virtual Control System

We install the VMware vSphere v4 on all of the physical machines and build a virtual cluster system. On the virtual machine system, we created some storage group for virtual machine for different type of application in centralized storage, typically one special small IOC storage group for all virtual IOC servers, two big storage group for NFS server and the Archive server. Before the storage group created, we built the virtual machine templates and installed the EPICS and application software with system configure of SSRF control system environment, include the EPICS environment, network with VLAN setting. Finally we create all of virtual server in the current SSRF control system from the template, include the Yum, NFS, NIS, Boot, Archive, Database, IOC and so on.

Testing of virtual control system environment

We have tested the virtual control system environment at the SSRF machine study time successful. All the functionality of control system environment can be operated normal and virtual IOC servers also work fine

with large numbers of network data transfers through the virtual network adapter, the maximal number of packets can reach about ten thousand per second. Except the mainly target of the virtual machine system, we also

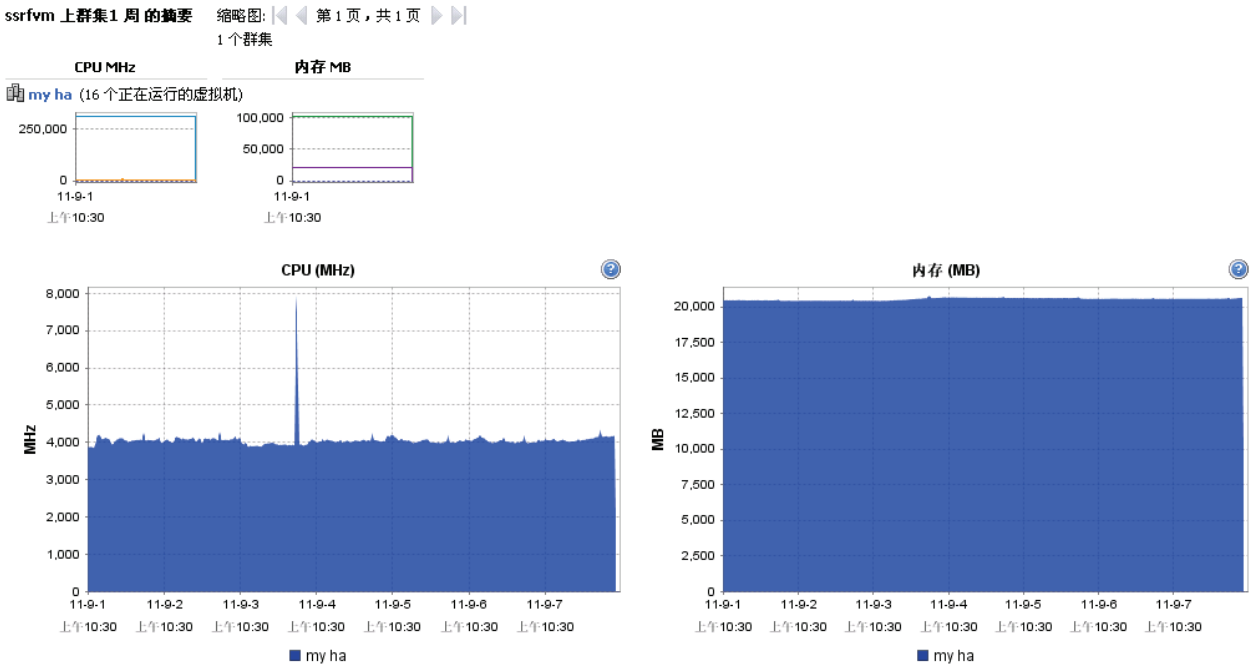


Figure 3: The resource average utilization of virtual machine cluster in one week.

tested the NFS and Archive disk read/write speed and all of results indicated the good performance of the virtualization, Fig. 3 shows the CPU and memory utilization in one week period. Include all the virtual server and application system, the online migration and physical machine crash simulation were tested and the fault migration ware successful. The virtual machine and application all can be moved from the crashed physical machine it existed to other normal physical machine.

During the running in couple of month, we met some of hardware failure situation such as the hard disk and network fault, even cause the physics server crashed, all the virtual servers were still work continually and the virtual control system environment also keep normal running. After fixed the physical server, it can add to the virtual cluster quickly and seemed nothing happened.

THE BENEFIT

The control system environment virtualization can integrate the hardware resource and raise the hardware utilization such as CPU and memory. The most direct results are we saved the servers and consume of power through the virtualization, we reduce the numbers of server in SSRF control system from more than 40 to 16.

The most import point is it can large improves the reliability of the control system environment and the MTBF of the whole facility by the high availability (HA) and VMotion characteristics of virtual machine system.

Other benefit is we can deploy virtual machine from the preconfigured virtual machine template easy and rapidly, that's bring more flexibility for the control system environment, all the work are the configuration of virtual system and no more any hardware work.

CONCLUSION

We have built a virtual control system environment with high availability and auto migration on fault characteristics and used in normal operation of SSRF successfully. From the beginning to now, it works more than half of year and got the result of very stable operation with good improvement of MTBF for control system.

Used virtualization for EPICS IOCs of many asynchronous devices and application server in SSRF control system got the more flexibility and stability.

The virtualization bring benefits such as save hardware resource and achieve the high availability goal and demonstrated the virtualization technical that applied in the larger scale control system environment.

REFERENCES

[1] http://www.vmware.com/pdf/virtualization.pdf
[2] http://www.ssrf.ac.cn
[3] Doug Murray*, Garth Martinsen, Judy Wang, A development environment for the SSC control system, Nuclear Instruments and Methods in Physics Research Section A Volume 352, Issues 1-2, 15 December 1994, Pages 390-392 CONTROL SYSTEM
[4] T. Ohata#, M. Kodera, M. Ishii, M. Takeuchi and T. FukuiA VIRTUALIZATION OF OPERATOR CONSOLES ON BEAMLINE,ICALEPCS07
[5] M. Eguiraun1, J. Jugo1 and etc NETWORKED CONTROL SYSTEM OVER AN EPICS BASED ENVIRONMENT, IPAC10

MONITORING CONTROL APPLICATIONS AT CERN

F. Bernard, M. Gonzalez, H. Milcent, L. B. Petrova, F. Varela#, CERN, Geneva, Switzerland.

Abstract

The Industrial Controls and Engineering (EN-ICE) group [1] of the Engineering Department at CERN has produced, and is responsible for the operation of around 60 applications, which control critical processes in the domains of cryogenics, quench protection systems, power interlocks for the Large Hadron Collider and other sub-systems of the accelerator complex. These applications require 24/7 operation and a quick reaction to problems. For this reason the EN-ICE group is presently developing the Monitoring Operation of cOntrols Networks (MOON) tool to detect, anticipate and inform of possible anomalies in the integrity of the applications. The tool builds on top of Simatic WinCC Open Architecture (WinCC OA) [2] SCADA and makes usage of the Joint COntrols Project (JCOP) [3] and the UNified INdustrial COntrol System (UNICOS) [4] Frameworks developed at CERN. The tool provides centralized monitoring and software management of the different elements integrating the control systems like Windows and Linux servers, PLCs, applications, etc. Although the primary aim of the monitoring tool is to assist the members of the EN-ICE Standby Service, the tool may offer different levels of detail, which also enables experts to diagnose and troubleshoot problems. In this paper, the scope, functionality and architecture of the tool are presented and some initial results on its performance are summarized.

INTRODUCTION

The LHC accelerator complex and its associated Experiments rely on many critical auxiliary systems for their safe operation. The EN-ICE group at CERN develops solutions and provides support in the domain of medium and large control systems covering the Experiments, as well as the technical infrastructure and accelerator systems. The group was born in 2009 following a major reorganization of the controls groups at CERN, which aimed at centralizing in a single group the experts and the knowledge on industrial control systems existing at CERN. EN-ICE currently provides an ample portfolio of solutions and actively participates in the development of two successful controls frameworks at CERN that build on top of the WinCC OA (formerly PVSS): the JCOP and the UNICOS frameworks. The utilization of these two frameworks is promoted by the group in all WINCC OA-based applications in order to reduce the development and maintenance efforts.

Moreover, the group also develops and maintains a large variety of turn-key applications in various fields like cryogenics, power interlock systems and safety. These applications are wide-spread around the CERN facilities and although many of them are operated by separated groups, EN-ICE is the ultimate responsible for their correct operation and maintenance over the lifetime of the LHC.

A quick response to abnormal situations or misconfiguration of the systems is crucial to maximize the physics usage of the LHC. Two main actions were taken by EN-ICE to ensure the maximum availability of these control applications, namely:

- A standby service was put in place to guarantee the availability of an expert 24/7.
- A tool was developed to centrally manage software upgrades and to monitor the integrity of the applications, which provides an efficient troubleshooting strategy to both, members of the standby service and application developers. This is the Monitoring Operation of cOntrols Network (MOON) tool and it is described in the rest of this contribution.

MOON

Why Yet Another Monitoring Tool at CERN?

In the last few years, a wide spectrum of monitoring tools has flourished at CERN. Although these tools provide many similar functionalities, this variety is, to some extent, justified by the peculiarities of the applications that are imposed by the technologies used, and the operational environment. Prior to deciding on the development of a new tool, an evaluation of various existing monitoring applications was carried out. The following were some of the main criteria that could not be satisfied by the tools evaluated and that led to the development of MOON:

- Integrated software configuration management and application monitoring allowing for quick correlation between mis-configuration of the monitored applications (e.g. wrong versioning of a component or configuration parameters) and run-time anomalies.
- Centralized deployment of software components on sets of WinCC OA-based applications.
- Detailed light-weight access to the run-time databases of the remote WINCC OA-based applications exploiting the native interfaces and protocols of the product.
- Need to combine in a single tool the monitoring of all elements integrating control applications, namely: hosts, WINCC OA-based applications and their associated front-end devices, like PLCs and FIP [5] buses, as well as the supervision of the technical infrastructure, e.g. electrical distribution, status of the network devices, etc.
- Graphical representation of the components of the control application and of their interconnections and dependencies.

#fernando.varela.rodriguez@cern.ch

- Support of multiple views presenting different levels of detail of the applications depending on the role of the user, e.g. a standby-service user with a limited knowledge of the applications as opposed to an expert or a developer.
- Multiplatform: given the extensive usage of Microsoft Windows and Linux for the control applications at CERN, detailed monitoring of hosts running these operating systems is required.
- Minimization of the development and maintenance efforts.
- Alarm system to list the current faults affecting the monitored systems. In addition the fault history to detect the recurrent problems.
- Long term storage of alarms and monitored parameters.
- Trending of online and historical values.

Moreover, it is very important to underline that MOON is not a complete new development as it is largely based on existing tool and reuses many components of the EN-ICE Frameworks as it is explained in the next section.

Architecture

MOON is an integrated monitoring and software management tool based on WINCC OA which exploits the following two main components of the JCOP Framework, which were originally developed for the control systems of the LHC Experiments:

- The Component Installation Tool [6] for centralized deployment of software components onto sets of WINCC OA-based applications.
- System Overview Tool [7] for farm and application monitoring.

The main building blocks of the application are shown in Figure 1. MOON extends the functionality of the JCOP System Overview Tool to also provide the monitoring of the PLCs and FIP agents. In addition, the demanding requirements of MOON in some specific areas like alarm handling and nightly reports called for a number of enhancements to the original functionality of the components that were successfully ported back to the base utilities. Moreover, the utilization of the UNICOS tools and concepts for the graphical interfaces provides an intuitive and powerful navigation schema for users with different technical backgrounds.

Functionality

The main graphical interface of MOON is shown in Figure 2. The panel is organized into multiple tabs that containing tree-like views of the applications and monitored equipment. These trees are implemented using the JCOP Finite State Machine (FSM) [8], which models each monitored component according to a well-defined set of states and possible transitions amongst them. Moreover, the FSM toolkit organizes these applications in a hierarchical fashion where the state of a node is calculated based on the state of its children where the leaves of the tree represent the actual monitored equipment, i.e. the PLCs, the hosts, the WINCC OA projects, etc.

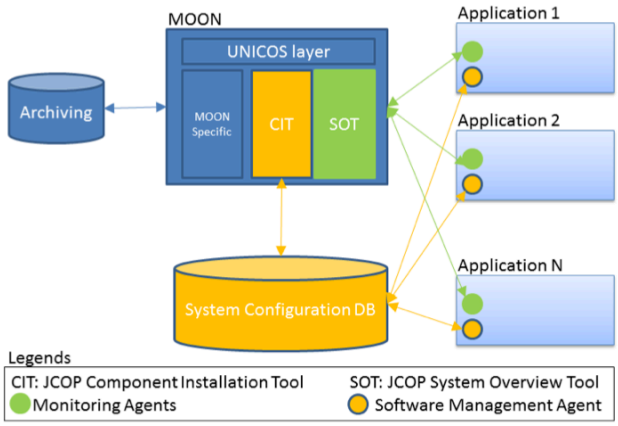


Figure 1: MOON main building blocks.

The state of each node in the tree is characterized by an associated colour. The propagation of the states and associated colours upwards in the FSM hierarchy summarizes the overall state of all monitored applications in a view and provides an intuitive method to locate the components of the control systems experiencing anomalies.

The selection of a node in a tree causes a dedicated panel to be displayed on the right-hand side of the graphical interface. The information displayed in those panels depends on the mode of operation of the tool: run-time monitoring or configuration management. Figure 4 shows an example panel in the former mode where a synoptic view providing a graphical representation of the monitored components, as well as their dependencies and connectivity is shown. In the configuration management mode, the panels display and allow modifying the configuration of the remote applications in a centralized fashion (Figure 5). This mode is restricted to experts by means of a strict access control policy and it is heavily used during technical stop periods of the LHC to upgrade sets of components in groups of applications. The arrangement of the applications in the FSM tree is preserved when toggling between the two modes of operation of the tool.

Each FSM view shows a different level of detail and logical organization of the equipment. As an example, the main view presented to the members of the standby service contains a coarse and simplistic representation of the control applications (e.g. current state and basic information of a device) whereas the expert view holds all required information for an expert to understand the behaviour of the application (e.g. statistics, performance counters).

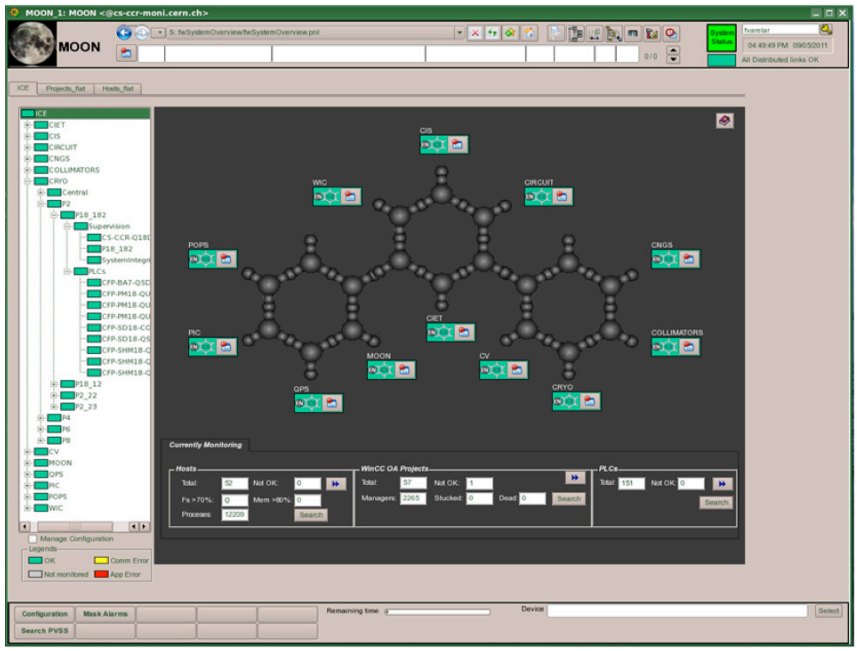


Figure 2: Main graphical interface of MOON showing a summary of the states of the EN-ICE controls applications.

At each level of the trees corrective actions on the remote applications can be taken by experts, e.g. to restart a process or reboot a host.

Problem identification is also complemented by the powerful alarm-handling schema featured by the tool. Besides an overall alarm screen, at each level of a FSM tree, the user can access a preconfigured alarm panel that restricts the alarms displayed to those raised by the equipment and applications in the local FSM sub-tree. Data gathered and alarms raised by MOON are archived to an ORACLE database for offline analysis. Alarms triggered by misbehaving sensors or problems in the readout can be masked by experts. This feature can also be used to prevent alarms from a particular set of applications during planned interventions. In this case, experts may define an expiration date for the masking such that the tool automatically unmasks these alarms at the due date.

MOON also provides a direct link from its alarm screen to the web help procedures written by experts that are associated with an alarm. Moreover, the tool also provides users with access to an electronics logbook to report an incident.

Web reports are generated daily for each application. These reports contain detailed information of the state of the different components integrating the control applications during the last 24 hours. In order to help analysing errors and predicting possible faults, the reports also include statistical data, like the number of occurrences of particular alarm.

System Size and Performance

The system which is currently in production has a considerable size and it is expected that it will grow in the near future (see Table 1). Currently, the parameters monitored by MOON are refreshed every few tens of seconds. However, this interval may be increased as system grows in size. Figure 3 shows the distribution of incidents per application domain over a week. Despite of the high number of incidents detected by the tool, an intelligent filtering schema is applied so that the standby service is notified only in the event of major faults.

Table 1: Monitored Items

Item	Current number	Estimated number	Refresh interval
Application Domains	12	20	N/A
Hosts (cores)	52 (544)	100	30s
PLCs	151	400	30s
WinCC OA applications	57	300	30s
Processes	12177	25000	30s

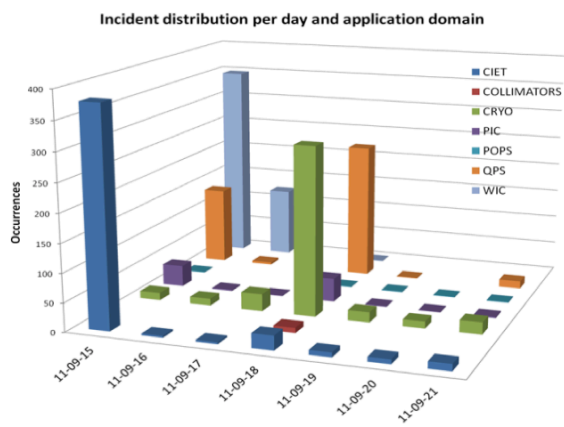


Figure 3: Incidents distribution.

NEXT STEPS

Currently, MOON monitors and provides views for the standby service for most of the EN-ICE WINCC OA applications in the accelerator and technical infrastructure domains. The development plan foresees a continuous upgrade of the tool with an incremental addition of new functionality whilst avoiding any interference with the monitored applications. In particular, a major milestone has been set for the unusually long stop scheduled at the end of 2011 beginning of 2012. During this period, it is planned to complete most of the functionality envisaged for the tool and to extend the range of monitored applications to Experiments. Specific actions include the completion of the expert views, enabling email notifications in the event of alarms, as well as the monitoring of devices and peripheral technical infrastructure, such as network equipment and electrical distribution that, although they are not a direct responsibility of EN-ICE, are vital for the correct operation of the control applications. Moreover, an effort will be made to extend the monitoring to the LabVIEW - based applications developed by the group. Finally, EN-ICE also foresees to offer the tool as a service to other groups at CERN having similar requirements in terms of application monitoring.

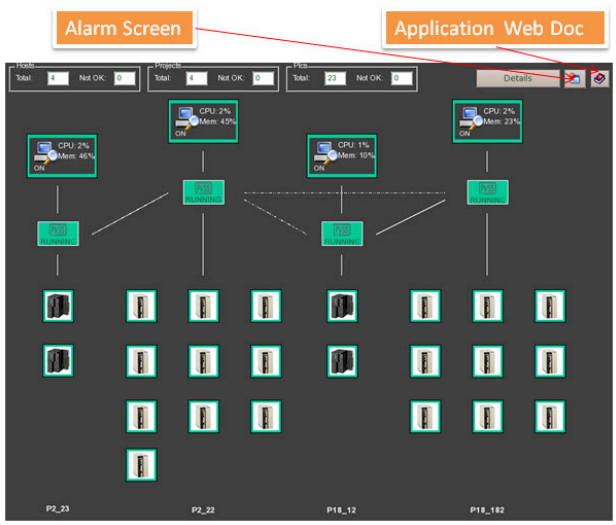


Figure 4: MOON user interface showing the layout of the cryogenics control applications for Point 2 of the LHC.

CONCLUSIONS

The modular design approach of the EN-ICE Frameworks, the clear definition of the user requirements gathered at the initial phase of the project, as well as the consistent model of abstraction the control applications has enabled a rapid development of MOON with limited resources. Although the tool has only recently entered the production phase, it has already shown its power to assist experts and members of the standby service in their daily work. A number of centralized upgrades have been successfully performed using the tool leading to a significant reduction of the time required for these interventions thus increasing work efficiency. Moreover, MOON has now become the primary tool for the members of the standby service to understand the behaviour of the control applications. The tool has successfully operated with no interference on the monitored applications since its initial deployment. For these reasons, MOON is already contributing its grain to maximize the Physics usage of the LHC by reducing the reaction time in the event of problems in the EN-ICE applications that control critical processes for the operation of the machine and the experiments.

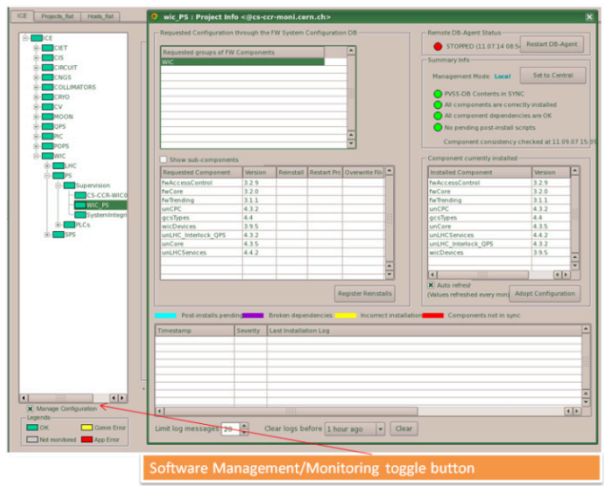


Figure 5: MOON software management panels showing the requested and current configurations of a WINCC OA project.

ACKNOWLEDGEMENTS

The authors would like to thank M. Boccioli, B. Copy, D. Galli, P. Golonka and C. Tamper for their contributions to MOON.

REFERENCES

[1] http://www.cern.ch/enice.
[2] http://www.pvss.com.
[3] O. Holme et al., "The JCOP Framework", ICALEPCS 2005, Geneva, Switzerland.
[4] Ph. Gayet, R. Barillere, "UNICOS a framework to build industry like control systems: Principles & Methodology", ICALEPCS 2005, Geneva, Switzerland.
[5] http://www.worldfip.org/.
[6] F. Varela, "Software Management of the LHC Detector Control Systems", ICALEPCS 2007, Knoxville, Tennessee, USA.
[7] M. Gonzalez, F. Varela, K. Joshi., "The System Overview Tool of the JCOP Framework", ICALEPCS 2007, Knoxville, Tennessee, USA.
[8] C. Gaspar, B. Franek, "Tools for the Automation of Large Distributed Control Systems", 14th IEEE Real Time Conference 2005, Stockholm, Sweden.

INFRASTRUCTURE OF TAIWAN PHOTON SOURCE CONTROL NETWORK

Y. T. Chang, C. H. Kuo, Y. S. Cheng, Jenny Chen, S. Y. Hsu, C. Y. Wu, K. H. Hu, K. T. Hsu
National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

A reliable, flexible and secure network is essential for the Taiwan Photon Source control system which is based upon the EPICS toolkit framework. Subsystem subnets will connect to control system via EPICS based CA gateways for forwarding data and reducing network traffic. Combining cyber security technologies such as firewall, NAT and VLAN, control network is isolated to protect IOCs and accelerator components. Network management tools are used to improve network performance. Remote access mechanism will be constructed for maintenance and troubleshooting. The Ethernet is also used as fieldbus for instruments such as power supplies. This paper will describe the system architecture for the TPS control network. Cabling topology, redundancy and maintainability are also discussed.

INTRODUCTION

Taiwan Photon Source (TPS) [1] will be the new 3 GeV synchrotron radiation facility to be built at National Synchrotron Radiation Research Center, featuring ultra-high photon brightness with extremely low emittance. The construction began in February 2010, and the commissioning is scheduled in 2014.

The control network is used for the operations of accelerators and beamlines. TPS control system will be implemented using the Experimental Physics and Industrial Control System (EPICS) [2] software toolkit. Control devices are connected by the control network and integrated with EPICS based Input Output Controller (IOC). The control network will be a 1-Gbps switched Ethernet network with a backbone at 10-Gbps.

INFRASTRUCTURE

The main goal of this planning is to build a reliable, agile and secure network for TPS control system. The design will provide enough flexibility and scalability for future expansion. [3]

Accelerator operators are the principal users of the control system. Control consoles with remote multi-display will be used to manipulate and monitor the accelerator through network. For remote monitoring and control Taiwan Light Source (TLS) facility, dedicated control consoles are planned to be installed in TPS control room.

Control System Computer Room contains EPICS control servers, database servers, control console computers, and network equipments. Network services will be available at the control room, control system computer room, 24 Control Instrumentation Areas (CIA), linear accelerator equipment area, transport lines, and main power supply equipment room which are distributed along the inner zone just outside of the machine tunnel. Each CIA serves for one cell of the machine control and beamline interface. Major devices and subsystems connected to the control system are installed inside CIAs.

Control network connects to NSRRC campus network through a firewall with Network Address Translation (NAT) function. Segregating the network will strengthen the security for those devices that need additional protection and high availability. Network traffic burden will also be lowered by isolating from general purpose network. Remote access mechanism will be constructed for maintenance and troubleshooting.

Connection to TLS control network is required for remote operations of the TLS facility. Control system laboratories for software development and hardware maintenance are also connected with the control network. The TPS control network infrastrcture is shown in Figure 1.

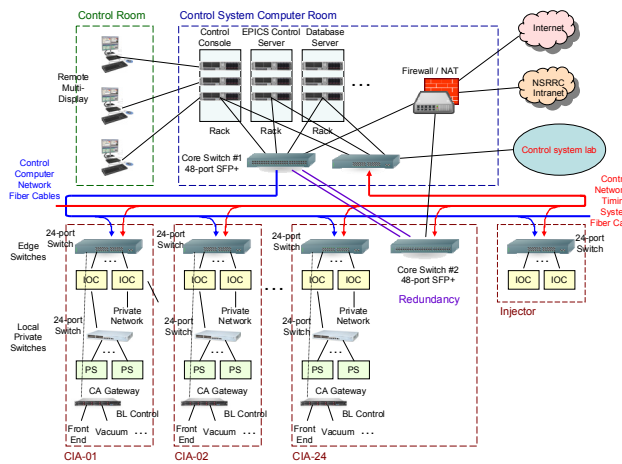


Figure 1: TPS control network infrastrcture.

A high performance switch with 48 10-Gbps fiber ports will be defined as the core switch. Two core switches will be used for redundancy, one is located inside the Control System Computer Room and the other is located inside CIA #24. Optical fiber links between core and edge switches are matched up with redundant cabling structure.

There are two types of switches be used in every CIA. The first type is defined as the edge switch which is used to connect IOC nodes and uplink to the high-speed backbone through 10-Gbps fiber uplinks. A 24-port 100/1000BASE-T switch with 2 10-Gbps fiber uplink ports wil be selected as the edge switch.

The second type is defined as the local private switch which is used for local private network to connect control devices such as power supplies and uplink to the IOC nodes. Depending on the needs, a variety of low cost Ethernet switches will be used as the local private switch.

Considering the budget, only one core switch will be used in Phase I. The redundancy structure will be implemented later in Phase II. But the fiber cabling for redundancy will be ready in Phase I. Figure 2 shows the baseline plan of the TPS control network.

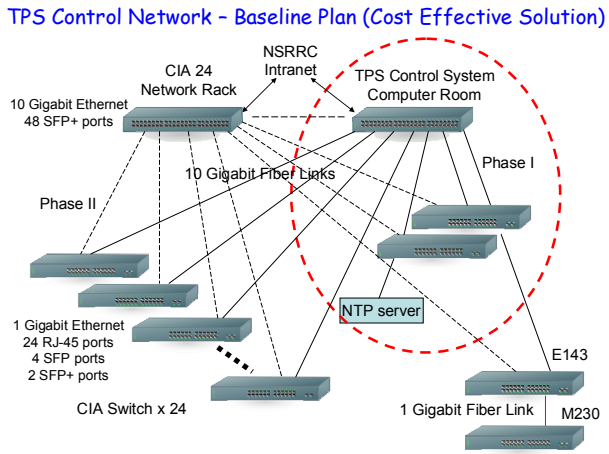


Figure 2: TPS control network – baseline plan.

SUBSYSTEM SUBNET

One Class B private network will be used for IOC network. Because there is a huge number of netwroked devices which scatter over a vast area (e.g. 24 CIAs), the IP addressing schema will be easy to identify the locations of IOCs and devices. This will be helpful to speed up finding the devices for maintenance and troubleshooting. This Class B private network will use IP range 172.20.xx.yy which xx represents locations (e.g. number of CIA) and yy is for functional groups.

There are multiple Class C private networks for respective subsystems, such as BPM IOCs, power supplies, motion controllers, GigE Vision, etc. These Class C private networks will use IP range 172.21.xx.yy which schema is also the same as above. Access of these Class C private networks might connect to the the VLAN router to provide access possibility.

Highly reliable Ethernet will be heavily used as fieldbus in the TPS control system. Power supplies for dipole, quadrupole and sextupole are connected to the EPICS IOCs by Class C private Ethernet within the CIAs.

Miscellaneous instruments will connect to the control system IOC located at each CIA via Ethernet also, such as LXI instruments, temperature/voltage monitors, etc. All of these devices might comply with LXI standard or not.

Orbit data is the most important operation information and should be captured in 10 Hz rate without interruption. In order to provide better service for this, a dedicated Class C subnet is planned for BPM IOCs. A CA gateway will connect with the BPM network to the TPS control network.

EPICS based CA gateway will provide necessary connectivity and isolation. Its functionality is to forward channel access to different network segments. It can also reduce network traffic and provide additional access security.

GigE Vision for diagnostics is based on the Internet Protocol standard and can be adapted to EPICS environment. Also, the images can be easily accessed through network for machine studies. The GigE Vision cameras can connect to control system through Ethernet with the data transfer rate up to 1000 Mbits/s. For decreasing traffic loading, one Class C private network and one CA gateway will also be used for the GigE Vision cameras to connect with the control system.

Subsystems such as vacuum, front-end, beamline control, and utility can access process variables of accelerator control system via CA gateways. It is expected that every beamline will have a Class C private network for their control system, data acquisition, and endstation applications. The beamline EPICS control environment will connect to the machine control system via CA gateway at each CIA. This design will provide necessary connnectivity between the machine control system and beamline control system and also restrict unnecessary network traffic across different network segments. The relations between the machine control system and beamline control do not clearly defined at current stage.

IP technology will be heavily used in the TPS control system. For providing more convenient environment for system maintenance, IP based cameras for area monitoring, phones, pagers are planned to attach to a Class C private network. This network will connect to the control network via a CA gateway and/or VLAN mechanism to the NSRRC intranet for saving network bandwidth.

For miscellaneous devices, the same principle will be adopted. The cabling scale will be downsized by using CA gateways and VLAN routing mechanism.

CABLING

For the long distance (> 100 m) networking, single-mode fiber (SMM) in the category G652.D will be used for cost consideration. There are two SMM cabling distribution links. One of the links is for control computer network only, the other is for control network and timing network dual function fiber pairs in one fiber cable.

The fiber cables for the control computer network will distribute from the Control System Computer Room and surround half of the ring clockwise and counterclockwise to every CIA. Then Copper STP/UTP cables are used to connect CIA edge switches to various IOCs and network attached devices within the same CIA. The fiber cabling for control computer network is shown in Figure 3.

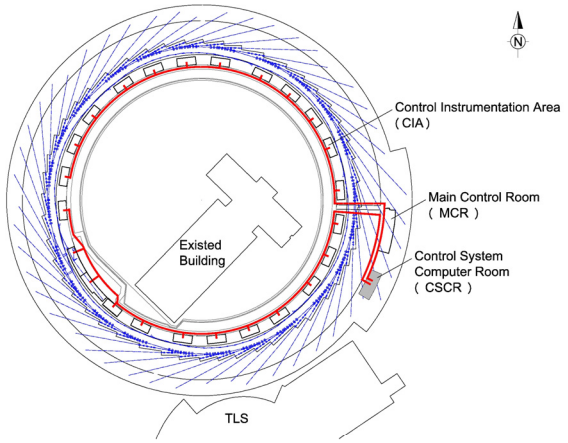


Figure 3: Fiber distribution for control computer network.

Since the timing master is located inside CIA #24, the fiber cables for control network and timing system will distribute from the CIA #24 to every CIA and Control System Computer Room. For receiving timing signals synchronously, the length of fiber cables will be equal. Besides, one fiber pair will be used as the redundancy for the control computer network. The fiber cabling for control network and timing system is shown in Figure 4.

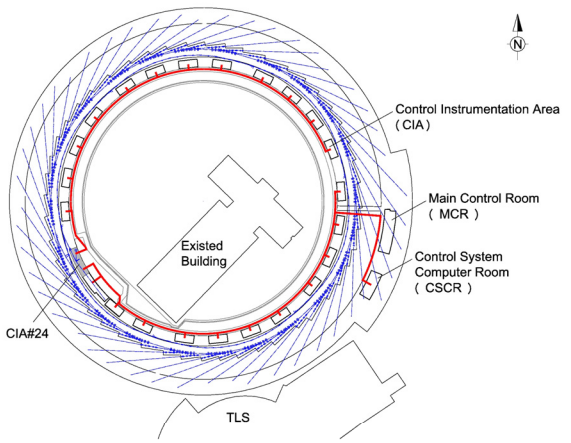


Figure 4: Fiber distribution for control network and timing system which strated form CIA #24.

NETWORK MANAGEMENT

Spanning Tree Protocol (STP) or Rapid Spanning Tree Protocol (RSTP) will be configured to implement redundancy. Network monitoring software (e.g. MRTG) will be used to show traffic and usage information of the network devices. By collecting and analyzing the packets, it can measure the traffic and usage to avoid bandwidth bottlenecks..

It is necessary to access the control system from outside in case of machine problems. Remote maintenance or troubleshooting has the advantages of convenience and time-saving. There are many ways to configure the network to enable remote access. Network tunneling tools, such as Virtual Private Network (VPN), can be used to penetrate the firewall system of the protected network. It can establish an encrypted and compressed tunnel for TCP or UDP data transfer between control network and public networks inside or outside the TPS. Providing a reliable authentication mechanism is also essential to remote access the control network.

The Network Time Protocol (NTP) servers are needed for timekeeping. NTP is used for synchronizing the clocks of computer systems over the TPS control network within 10 ~ 100 millisecond performance.

Using Simple Network Management Protocol (SNMP), the behaviour of network-attached devices can be monitored for administrative attentions. Since the TPS control system is based upon the EPICS framework, a dedicated EPICS IOC with SNMP support will be used to monitor the status of control system components such as CompactPCI (cPCI) IOC crates, network switches, servers, Uninterruptible Power Supplies (UPSs), etc. The equipment room environment such as temperature and electric power will also need to be watched. [4]

CYBER SECURITY

Current accelerator control systems are commonly based on modern Information Techonolgy (IT) hardware and software, such as Windows/Linux PCs, PLCs, data acquisition systems, networked control devices, etc. Control systems are correspondingly exposed to the inherent vulnerabilities of the commerial IT products. Worms, viruses and malicious software have caused severe cyber security issues to emerge.

It is necessary to use network segregation to protect vulnerable devices. Combining firewall, NAT, VLAN… technologies, control network is isolated to protect IOCs and accelerator components that require insecure access services (e.g. telnet).

Firewall only passes the packets from authorized hosts with pre-defined IP addresses outside control network and opens specific service ports for communications. But firewall is not able to resist the spread of worms. Worms are not only designed to self-replicate and spread but also consume the network bandwidth. Thus security gateway or IPS (Intrusion Prevention System) is needed to block worm attacks and quarantine suspicious hosts. IPS can detect and stop network threats such as worms, viruses, intrusion attempts and malicious behaviors.

Remote access mechanism needs network tunneling applications to bypass the firewall. It will provide a private tunnel through the public network for remote access to the control network. The remote access mechanism also requires appropriate types of protection and control. It must be enhanced with a reliable user authentication mechanism for full security.

Security will always put at the highest priority for the TPS control system. Security policy for control network is essential. Regulations should be defined for the accelerator scientists and engineers to access the control system. It's everyone's responsibility to protect the

infrastructure. However, balance between security and convenience will be addressed also.

SUMMARY

This report describes the infrastructure of the TPS control network. An adaptive, secure and fault-tolerant control network are essential for the stable operation of the TPS. The control network will be separated from the NSRRC campus general purpose network for imposing security. Subsystem subnets will connect to control system via CA gateways for forwarding data and reducing network traffic. Two fiber cabling distributions are described. Network management tools will be used to enhance productivity. Remote access mechanism with proper authentication will be implemented for system maintenance or troubleshooting. An infrastructure monitoring system is planned to adopt the EPICS and SNMP. Cyber security will be the most concern.

REFERENCES

[1]. TPS Design Book, v16, September 30, 2009.
[2]. Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/
[3]. Y.T. Chang, "Preliminary Planning of Taiwan Photon Source Control Network", ICALEPCS 2009
[4]. Y.T. Chang, "Plans for Monitoring TPS Control System Infrastructure Using SNMP and EPICS", PCaPAC 2010.

DISTRIBUTED MONITORING SYSTEM BASED ON ICINGA

C. Haen*, E. Bonaccorsi, N. Neufeld, CERN, Geneva, Switzerland

Abstract

The LHCb online system relies on a large and heterogeneous IT infrastructure: it comprises more than 2000 servers and embedded systems and more than 200 network devices. Many of these equipments are critical in order to run the experiment, and it is important to have a monitoring solution performant enough so that the experts can diagnose and act quickly. While our previous system was based on a central Nagios server, our current system uses a distributed Icinga infrastructure. The LHCb installation schema will be presented here, as well some performance comparisons and custom tools.

INTRODUCTION

LHCb [1] is one of the four large experiments at the Large Hadron Collider at CERN. The infrastructure of networks and servers deployed in order to manage the data produced by LHCb and control the experiment are critical for its success. While for the control and monitoring of detectors, PLCs, and readout boards an industry standard SCADA system PVSSII has been put in production, at a lower level the network infrastructure and resources used by each server in the LHCb Cluster need to be monitored. This is because the PVSSII in the ECS (Experiment Control System) depends already on a lot of systems such as the shared storage, the network, DNS (Domain name Server) and others. In monitoring terminology, one uses 'host' to refer to a machine such as a server, and 'service' to refer to all kind of software, application or resource like CPU or disk space, etc. A 'service' is then applied to an 'host'. Over time, the amount of services and hosts to be monitored increased up to the point where the previous monitoring system shown on Fig. 1 based on Nagios [2] did not perform sufficiently, and we decided to search for a more scalable solution.

INSUFFICIENT PERFORMANCES

Nagios monitors hosts and services with tiny single-purpose programs called plugins that are executed periodically in order to get the status of a monitored entity [3] (host or service). Basically, executing a plugin consists in forking, creating and closing a socket. Thus, monitoring 40 000 entities induces a huge load on the central server running Nagios [4]. The result is a big latency in the checks and then in the alerts received by the experts.

* christophe.haen@cern.ch

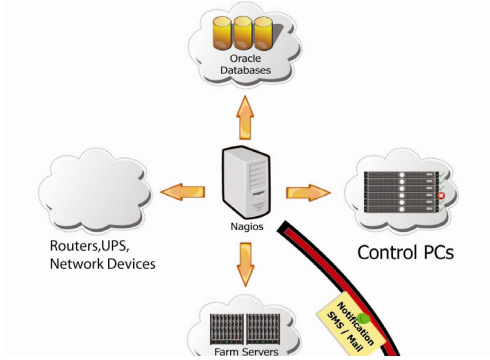


Figure 1: The previous Nagios installation.

NEW INFRASTRUCTURE

In order to improve our system, two changes took place: replacing Nagios by Icinga [5]; and going from a central to a distributed architecture.

Icinga is a fork of Nagios. There are several reasons to motivate the choice to switch:

- The support of the Icinga community is better.
- Extra features [6]: among them, the possibility of using a database as backend which can be queried with an API. This is particularly interesting regarding some other projects on going at LHCb.
- Nagios configuration files are compatible with Icinga.

Going for a distributed system is an obvious solution to avoid the bottleneck presented in the previous section. Three different possibilities to distribute our monitoring have been explored.

Independent Icinga Instances

The first option is to have several independent Icinga instances running, each of them monitoring a specific group of entities. The major drawback of that solution is the difficulty to have a redundancy in the checks: having the same checks executed by different instances would mean having all the alarms several times, since there is no information sharing between the instances.

Central Instance with Distributed Instances

In this setup, several instances would monitor specific entities, but they would all report to a central instance, whose goals are only to gather the check results, display it to the users, and trigger the notifications. This setup is possible with Icinga core [7]. If one of the distributed instance is dead, the central server will actively do the test,

thus ensuring redundancy for the distributed nodes. This solution has been seriously considered, but the configuration becomes complicated when you have many entities to monitor: the central server has to know all the entities, but each distributed server has to know only its own. In order to balance the drawback of this setup we developed a few tools to ensure consistency and dependencies:

- Oracle database backend: the database schema can be used to store a complete configuration for Icinga.
- Web frontend: based on symfony [8], this tool provides a global overview of the configuration, and allows non experts to use the monitoring services, by defining their own entities.
- Generation tool: thanks to a set of perl scripts, the configuration is extracted from the database, written in files readable by Icinga, and properly shared between the Icinga instances.

Even by using these tools, the management of large amount of entities was difficult: the frontend performances were not good enough to cope with our large installation. For this reason, we decided to give up that solution.

Central Icinga Instance with Distributed Workers

The third option is to have a single instance of Icinga which would schedule the checks and deal with the results, but having other servers (called 'workers') actually perform the checks. This is made possible by mod_gearman [9], an Icinga/Nagios broker module. The load induced on the worker is negligible, whereas the central server is fully busy. Mod_gearman is a module based on Gearman [10], a generic framework to distribute applications. It uses a client/server model as shown in Fig. 2. The server manages queues that clients use to get their tasks and give their results.

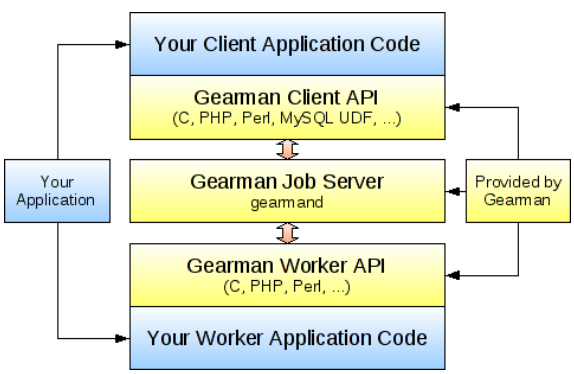


Figure 2: Gearman Stack.

Mod_gearman intercepts the checks triggered by Icinga, and puts them into queues managed by Gearman. The client side is a light weight program which simply gets the path of the executable, runs it, and put the result back in a result queue on the server side. Note that the executables need to be present on all workers. The elements in the result queue are then passed to Icinga, which processes them as if it had executed the checks itself.

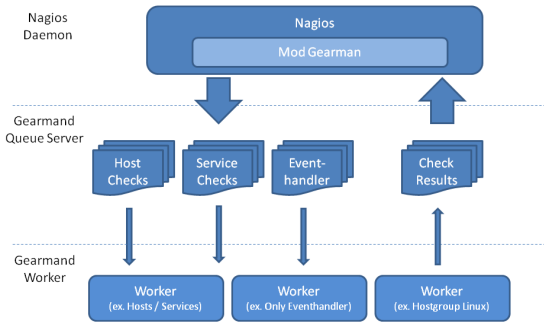


Figure 3: Mod gearman.

By default, all the checks are put in the same default queues showed in Fig. 3. This can be a problem if some workers cannot access the machine they should check, for network security reasons for example. To solve this issue, mod_gearman offers to put checks in separate custom queues, according to the group they belong to. These groups are groups of hosts or services defined in the Icinga configuration files: one can define one group for each separate network, and configure workers to fetch checks only from appropriate queue.

This is the solution we decided to implement in LHCb, and coupled it with a configuration schema described in the next section.

The other interesting feature proposed by Icinga is the database backend. With the idomod and ido2db modules [11], Icinga will log the result of every action and check-result in a database. The information can then be queried using a PHP API. This is a remarkable tool to use the Icinga results as input for other projects. Unfortunately, we found a deadlock in ido2db, for which a ticket has been open, which make it unusable for the moment.

Finally, to make sure we can always access our monitoring services, the server running Icinga is connected to the LHCb network and to the CERN network. Thus, even if a big network problem occurs in LHCb, the monitoring information can still be accessed through independent networks.

NEW SCHEMA

The whole LHCb infrastructure is made up of thousands of machines. We can divide in groups which are doing the same tasks, and thus must be monitored the same way.

Rewriting all the time the same complete configuration file for the different machines would be extremely tedious and error-prone. Fortunately, one can use two features offered by Icinga to avoid that:

- Hostgroup: an hostgroup simply defines a group of hosts. A host can be in several hostgroups, and a hostgroup can be member of another hostgroup.
- Inheritance: a host can inherit the configuration of another host, in the manner of the object oriented programming paradigm. The inheritance will only carry over the host monitoring options, not the services applied on the parent host. For example, the child will inherit the check frequency from its parent, but the service checks that are applied to the parent will not be applied to the child.

Thus by defining precisely the roles of all the machines, one can come up with a complete logical tree like in Fig. 4, which is then implemented as two separate but homeomorph trees. The first one is made of hostgroups: each node is member of the parent node. The service checks are always applied to a hostgroup in this tree, and never to a single host. The second tree is implemented with hosts, and each node inherits from its parent node: each level brings more specialized options, and the leafs of the tree are the machines themselves.

- Adding a new host consists in defining a name and an address, assigning it to the right hostgroup and inheriting from the right template (both being at the same in the logical tree). It will then be monitored exactly the same way as all the machines having the same role.
- Changing the way a functional group of machine is monitored is done by changing a single configuration file: for example, one can add a new service check on all of the file servers by adding this check only once to the hostgroup containing all the file servers.

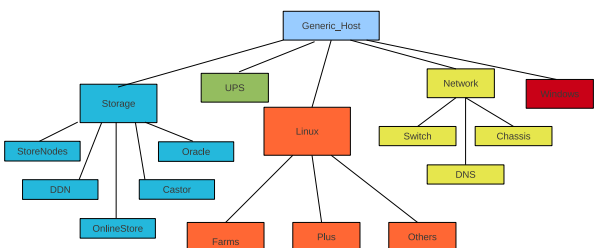


Figure 4: Part of the tree used to define configuration.

NEW PERFORMANCES

The new installation shown in Fig. 5 is now running 36000 service checks on 2100 hosts, with 50 gearman workers, and the performances are now on average 2760% better, as shown in Tables 1 and 2:

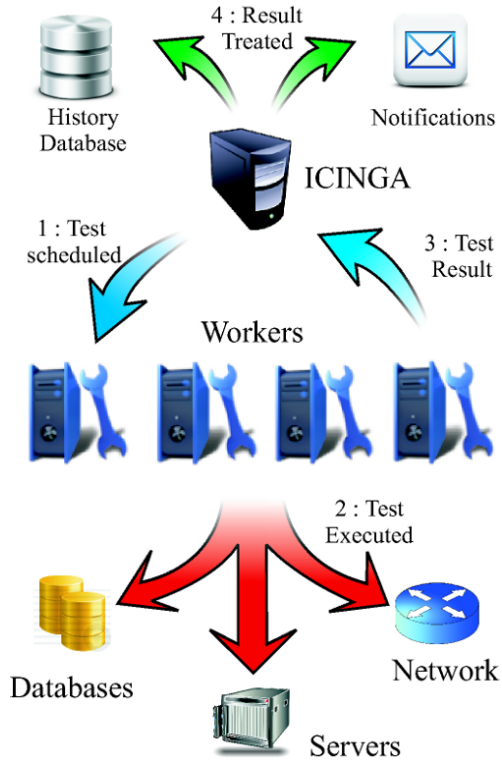


Figure 5: The new LHCb Icinga infrastructure.

Table 1: Single Nagios Instance Performances

	Min.	Max.	Average
Service checks latency	320 sec	578 sec	328 sec
Host checks latency	0 sec	401 sec	318 sec

Table 2: Central Icinga Instance and Distributed Workers Performances

	Min.	Max.	Average
Service checks latency	0.03 sec	57 sec	14 sec
Host checks latency	0 sec	35 sec	12 sec

The latency represents the difference of time between the scheduled execution time of the check, and the actual execution time.

CONCLUSION

Icinga has been running for two months in parallel with Nagios. The notifications have always been quicker and more up to date. We retired Nagios a month before this paper has been written.
In addition to that, the way our workers are spread over

the LHCb network, and the way the central Icinga server is connected to the external network ensures to have in almost any case access to our monitoring information.
Nevertheless, the central Icinga instance is still a single point of failure. A solution for this could be to set up a cluster of identical machines running in active-passive mode: if the server running the central Icinga instance crashes, a fail-over will happen and another server will take over.

REFERENCES

[1] A. Augusto Alves et al. The LHCb Detector at the LHC. JINST, 3:S08005, 2008.

[2] http://www.nagios.org/about

[3] W. Barth "Nagios, Systems and Network Monitoring" (2006)

[4] E. Bonaccorsi, Niko Neufeld "Monitoring the LHCb experiment computing infrastructure with Nagios" TUP001 ICALEPCS 2009

[5] https://www.icinga.org/about/

[6] https://www.Icinga.org/Nagios/

[7] http://docs.Icinga.org/latest/en/distributed.html

[8] http://www.symfony-project.org/

[9] http://labs.consol.de/lang/de/Nagios/mod-gearman/

[10] http://gearman.org/

[11] http://docs.Icinga.org/1.3.0/en/db_components.html

EFFICIENT NETWORK MONITORING FOR LARGE DATA ACQUISITION SYSTEMS

D.O. Savu, B. Martin, CERN, Geneva, Switzerland
A. Al-Shabibi, Heidelberg University, Heidelberg, Germany
R. Sjoen, University of Oslo, Norway
S.M. Batraneanu, S.N. Stancu, UCI, Irvine, California, USA

Abstract

Though constantly evolving and improving, the available network monitoring solutions have limitations when applied to the infrastructure of a high speed real-time data acquisition (DAQ) system. DAQ networks are particular computer networks where experts have to pay attention to both individual subsections as well as system wide traffic flows while monitoring the network. The ATLAS Network at the Large Hadron Collider (LHC) has more than 200 switches interconnecting 3500 hosts and totaling 8500 high speed links. The use of heterogeneous tools for monitoring various infrastructure parameters, in order to assure optimal DAQ system performance, proved to be a tedious and time consuming task for experts. To alleviate this problem we used our networking and DAQ expertise to build a flexible and scalable monitoring system providing an intuitive user interface with the same look and feel irrespective of the data provider that is used. Our system uses custom developed components for critical performance monitoring and seamlessly integrates complementary data from auxiliary tools, such as NAGIOS, information services or custom databases. A number of techniques (e.g. normalization, aggregation and data caching) were used in order to improve the user interface response time. The end result is a unified monitoring interface, for fast and uniform access to system statistics, which significantly reduced the time spent by experts for ad-hoc and post-mortem analysis.

INTRODUCTION

At the core of the ATLAS DAQ infrastructure there are 3 distinct computer networks responsible for the data transfers between the system's subcomponents. More than 200 switches and routers interconnect around 3500 hosts to build a real-time filtering system for particle collision events.

The ATLAS Networking Team's operational goals are to prevent network downtime and to be able to track down ad-hoc or post-mortem network issues as fast as possible. A complex software solution has been developed to help networking experts accomplish their goal while providing relevant and up-to-the-minute system information for other related DAQ teams.

The network monitoring software is designed as a modular solution around a central database (N-CDB). The N-CDB acts as a shared data structure for the various modules and is implemented as a core relational database with external binary file extensions. The most important software modules (Figure 1) are:

- **Topology Consistency:** responsible for keeping an up-to-date database representation of network devices, computers, connections and geographical location;
- **Statistics Collection:** gathering any network traffic related statistics and making the data available to other modules through the N-CDB;
- **The ADAM API:** a programmatic way of accessing the database data by external programs or scripts;
- **The User Interface:** a set of web-based applications to make all the information available in a structured and easy to navigate way;
- **Self-check module:** a warning mechanism sending detailed mail messages to experts about any module not functioning as expected.

TOPOLOGY CONSISTENCY

An accurate representation of installed devices and network connections in the N-CBD is assumed by every module and is critical for the system's operation. To keep the topology representation up to date, a full network discovery process is run periodically and is complemented by additional data. The discovery is then deployed in the production system through a semi-automatic procedure involving expert approval for topology changes.

The network discovery is primarily MAC address based, being able to identify network device uplinks and end node connections by inspecting the MAC address tables of network devices. Since the MAC address tables are dynamically populated by the traffic traversing network devices, it is essential to run the discovery when all the devices have generated traffic. Thus it is most efficient to run a discovery either during data taking, when almost all devices have network activity, or immediately after an induced ARP* broadcast event.

A network discovery based on MAC address tables or LLDP[#] does not reveal the nodes' function or geographical location. To extend the discovery knowledge with such complementary data, information from external databases is used and cached by the N-CDB. The purpose of the N-CDB cache is to avoid any run-time dependency and overloading of external databases.

*Address Resolution Protocol
[#] Link Layer Discovery Protocol

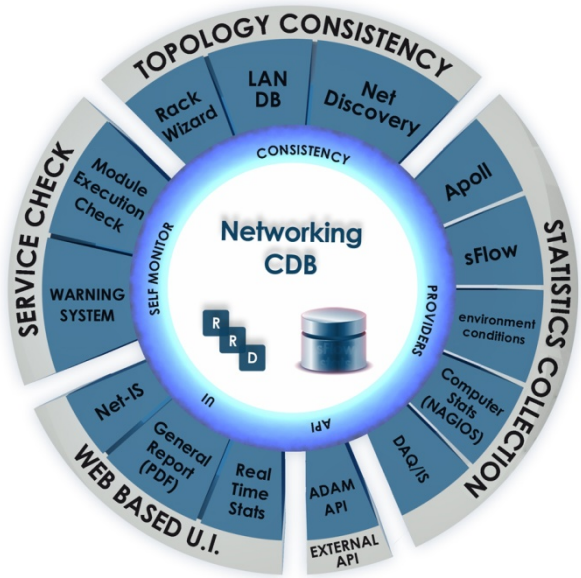


Figure 1: Diagram of the network monitoring software modules.

Such external databases include Atlas RackWizard, for geographical location, Sysadmin ConfDB, for functional host name mappings and IT LanDB, for auxiliary host mapping.

STATISTICS COLLECTION

Gathering statistics about traffic flows and network state is mandatory for monitoring the system's performance. The network overview statistics are the primary source of information for investigating most network problems. These statistics are collected by APoll, an internally developed software which polls SNMP counters from switches and routers every 30 seconds. Once a network issue is isolated, an in-depth analysis is performed by looking for relevant traffic samples collected via the sFlow protocol.

APoll, the high speed SNMP poller, has been developed by the networking team to address the limitations of commercial software used in production. It is implemented as a multi-threaded C++ application that dynamically adapts its number of threads to the number and responsiveness of the network devices. The run mode can be configured either as standalone, dumping basic traffic log files, or integrated mode when last minute statistics are synchronized with the N-CDB.

The SNMP* counter statistics, due to their time-series format, are stored in a structured set of Round Robin Database (RRD) binary files. The files are referenced by the metadata in N-CDB and accessed directly via the RRD library. A copy of the RRD files for the last 72 hours is also stored on a ramfs partition to reduce the I/O to non-volatile storage and improve application response. Additionally, the last SNMP counter values are stored in a N-CDB in-memory table. The table is used as a shared resource between all the applications that need or provide real-time data.

The sFlow sample data, on the other hand, does not resemble a standard time-series format. To efficiently store the sFlow gathered data, a tool is used to aggregate it over 1 minute time intervals and store it in a MySQL relational database. Then, various database tuning techniques are used to improve the response time.

Sometimes external factors, such as environmental conditions or faulty systems, affect the network's normal behaviour. Making information from directly related systems available to network monitoring modules gives the networking team an advantage in understanding and limiting the impact of an external event. Currently, full or partial information from NAGIOS (computer monitoring), PVSS (environmental conditions database) and DAQ/IS (data-taking information service) is accessible through the same user interface used for network monitoring. Examples of such external factors include rack power failure causing device crash events or DAQ data-taking process causing network discarded packets.

THE ADAM API

The information stored in the N-CDB is also of interest to shifters and system analysts. To facilitate programmatic access to information, a generic interface for data exchange has been implemented. The creation of a generic interface has been a joint effort between several ATLAS teams and lead to the definition of the ADAM (ATLAS Data API Mechanism) data exchange interface.

The ADAM API is fully implemented in the network

* Simple Network Management Protocol

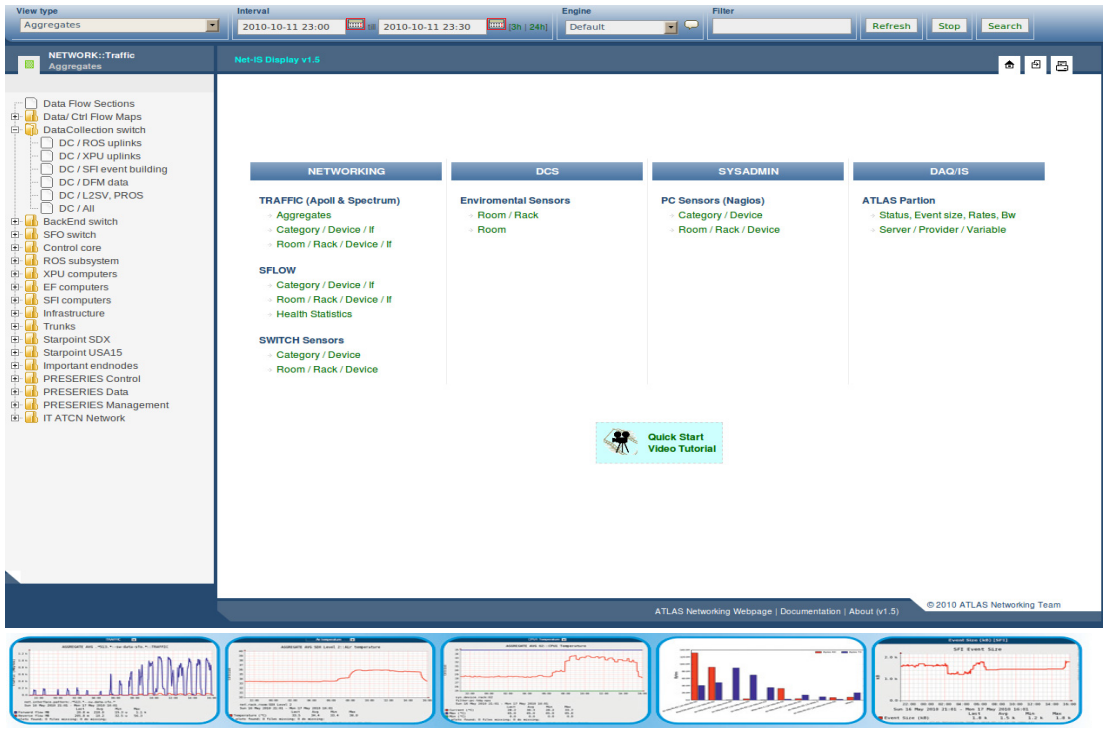


Figure 2: Main page of the Net-IS user interface providing access to historical statistics.

monitoring solution and provides network traffic, computer statistics and environmental conditions to external applications on demand. Through this API the network monitoring software can be a data provider for other external data analysis applications, and is currently used in the ADAM User Interface project.

THE USER INTERFACE

The user interface is designed to provide all the data an expert needs to investigate a problem and yet be simple to use for non experts. A set of web-based applications, integrated as a portal, offers access to real-time reports and historical statistics about network state and complementary systems. For offline visualization of the current topology a comprehensive PDF report is generated daily and made available for download.

The Net-IS web application (Figure 2) is the main networking interface used by experts and shifters. It provides direct access to any historical statistics through a single interface with the same look and feel regardless of the data source. The statistics can be grouped to match predefined or custom rules via an aggregation layer based on regular expressions.

The Net-RT web application provides real-time statistics about any network device or network traffic. Various table-based reports, as well as a 2D map displaying network uplinks load, are updated in real-time according to the N-CDB in-memory table.

A different option to access the N-CDB data is through a Command Line Interface (CLI). The CLI tool was developed to allow experts to access and change the N-CDB data manually without the risk of affecting its low level consistency.

The user interface has been described in more detail in the "Integrated System for Performance Monitoring of the ATLAS TDAQ Network" paper, published in the Proc. Computing in High Energy Physics 2010.

SELF-CHECK MECHANISM

The reliability of the system may be affected by a misbehaving module. The purpose of the self-check mechanism is to identify and inform experts in real-time about any faulty modules. To make the self-check mechanism more robust, the checking of module execution status and the mail reporting have been implemented as two distinct components.

The execution status check is implemented using a set of scripts that monitor each module's functionality and expected results, and then report the status to the N-CDB. If a check script fails to execute within a timeout period, it will flag this problem to the warning component. A module can also flag a warning and send a detailed message if it anticipates an imminent problem likely to require expert intervention.

The warning component then checks all the active module status information and sends error-triggered and daily service status mail reports.

CURRENT STATUS AND FUTURE PLANS

Started as an independent network monitoring software, the current ATLAS DAQ network monitoring solution has grown by integrating additional system information.

This extension improved the understanding of overall system behaviour by enabling the correlation between networking events and external factors. Two years after its deployment it is the main tool used by both experts and non experts to analyze network problems. It has proven to be more efficient compared to the alternate commercial solution when it comes to performance monitoring.

Future plans to improve the current solution include the extension of the N-CDB by storing network event logs from multiple sources and the addition of an event processing engine. Such a feature will bring together the network events signalling a state change, with the collected statistics that provide a more complete picture of the circumstances associated with the generated event.

REFERENCES

[1] DO. Savu, A. Al-Shabibi, B. Martin, R. Sjoen, SM. Batraneanu and S. Stancu, “Integrated System for Performance Monitoring of the ATLAS TDAQ Network”, in Proceedings of the CHEP 2010, Taipei, Taiwan, Oct. 2010.

[2] M. Ciobotaru, L. Leahu, B. Martin, C. Meirosu and S. Stancu, “Networks for ATLAS trigger and data acquisition”, in Proceedings of the CHEP 2006, Mumbai, India, Feb. 2006.

[3] S.M. Batraneanu, A. Al-Shabibi, M. Ciobotaru, M. Ivanovici, L. Leahu, B. Martin and S. Stancu, "Operational Model of the ATLAS TDAQ Network, Proc. IEEE Real Time 2007 Conference, Chicago, USA, May 2007.

[4] R. Sjoen, S. Stancu, M. Ciobotaru, S.M. Batraneanu, L. Leahu, B. Martin and A. Al-Shabibi, "Monitoring Individual Traffic Flows whithin the ATLAS TDAQ Network”, CHEP, Prague, Czech Republic, 21-27 Mar 2009.

[5] S. Kolos et al., “Online Monitoring software framework in the ATLAS experiment”, in Proceedings of the CHEP 2003, La Jolla, USA.

[6] W. Vandelli et al., “Strategies and Tools for ATLAS Online Monitoring”, in IEEE Transactions on Nuclear Science (TNS), June, 2007, volume 54, pp 609-615.

[7] SNMP, Simple Network Management Protocol [Online]. http://en.wikipedia.org/wiki/Simple_ Network_ Management_Protocol.

[8] WWW Technologies [Online]. Available: http://en.wikipedia.org/wiki/World_Wide_Web.

[9] Django, The Web framework for perfectionists with deadlines. [Online]. http://www.djangoproject.com

[10] RRDTool, OpenSource industry standard, high performance logging and graphing system. [Online]. http://oss.oetiker.ch/rrdtool/

[11] NAGIOS, The Industry Standard In IT Infrastructure Monitoring. [Online]. http://www.nagios.org

[12] LLDP, Link Layer Discovery Protocol. [Online]. http://en.wikipedia.org/wiki/Link_Layer_Discovery_ Protocol

VIRTUALIZATION FOR THE LHCB EXPERIMENT

E. Bonaccorsi, L. Brarda, M. Chebbi, N. Neufeld, CERN, Geneva, Switzerland
F. Sborzacchi, INFN, Laboratori Nazionali di Frascati, Italy.

Abstract

The LHCb Experiment, one of the four large particle physics detectors at CERN, counts in its Online System more than 2000 servers and embedded systems. As a result of ever-increasing CPU performance in modern servers, many of the applications in the controls system are excellent candidates for virtualization technologies. We see virtualization as an approach to cut down cost, optimize resource usage and manage the complexity of the IT infrastructure of LHCb. Recently we have added a Kernel Virtual Machine (KVM) cluster based on Red Hat Enterprise Virtualization for Servers (RHEV) complementary to the existing Hyper-V cluster devoted only to the virtualization of the windows guests. This paper describes the architecture of our solution based on KVM and RHEV as along with its integration with the existing Hyper-V infrastructure and the Quattor cluster management tools and in particular how we use to run controls applications on a virtualized infrastructure. We present performance results of both the KVM and Hyper-V solutions, problems encountered and a description of the management tools developed for the integration with the Online cluster and LHCb SCADA control system based on PVSS.

INTRODUCTION

LHCb is an experiment set up to explore what happened after the Big Bang that allowed matter to survive and build the Universe we inhabit today, in the specific is a dedicated heavy-flavour physics experiment designed to perform precise measurements of CP violation [1]. The experiment is located at point 8 of the LHC particle accelerator.

The LHCb online system has been designed to run completely isolated and independent, as an autonomous system, it consist of ~2000 physical servers and embedded systems interconnected through 3 main high density routers and ~100 distributions switches.

Hosts are organized in two different local area networks: the Experiment Control System (ECS) [2], illustrated in Figure 1 and the Data Acquisition System (DAQ). The access to the CERN General Purpose Network (GPN) and consequently to internet is provided by Linux and Windows gateways which are secured by a three tier firewall setup.

While the DAQ hosts have been designed to discern the "potentially interesting event" from the huge amount of data produced by the hadrons collisions, zero-suppressed in the front-end electronics [3], the ECS network has been designed to control the experiment, mainly using open sources software wherever possible and the standard LHC SCADA system PVSS in order to control and monitoring high and low voltages, gas and temperatures, etc.

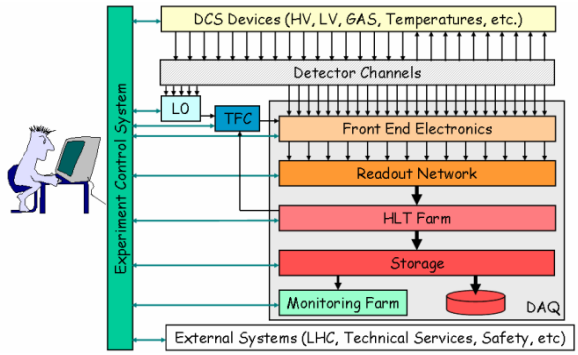


Figure 1: ECS Diagram.

Unlike the DAQ hosts the ECS hosts most of the time underuse resources (with some peak time to time) in terms of memory, network, power, cooling and space.

Taking into account that servers are becoming increasingly powerful, the use of the many-core CPUs accentuates this issue.

Virtualization has in principle a great promise for a control system like ours. It can save power and space and in the long run also money. At the same time it increases the availability and serviceability by abstracting software services from the underlying hardware. However, as we had to learn, the initial investment is rather high and many things have to be taken into account.

The LHCb online team has performed an evaluation of available clustered virtualization implementations focusing mainly on the free edition of Microsoft core Hyper-V [4].

The first part of the project was focused on the virtualization of the public web services and the essential infrastructure services summarized in Table 1.

In this paper we describe further work were we had add a virtualization implementation based on Linux KVM and Red Hat Enterprise Virtualization (RHEV).

Our plan is to migrate from the Microsoft Hyper-V infrastructure to the RHEV infrastructure as well as the deployment of virtual "experiment control PCs", in charge of controlling the detector hardware.

Table 1: Virtualized Systems

Category	Virtualized Systems
Public available	Web Services
Common infrastructure	Firewall, DNS, Domain Controllers, Cron system, DHCP
ECS	Control PCs
Test systems	Dedicated control PCs for testing software and procedures

HYPERVISORS

The hypervisor, also called virtual machine monitor, is a virtualization platform that allows multiple operating systems to run on a single host at the same time.

In the initial part of the study Microsoft Hyper-V had been chosen as hypervisor mainly because the virtualization technology offered by Red Hat/Scientific Linux was in a transition state from XEN to KVM.

A three nodes RHEV cluster has been tested and put into production in parallel with the Hyper-V cluster. The RHEV architecture is illustrated in Figure 2.

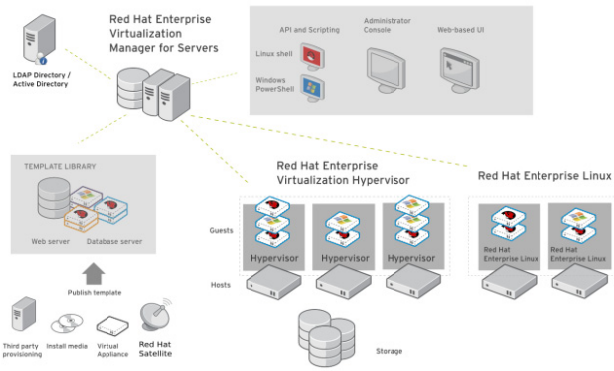


Figure 2: RHEV Architecture.

HARDWARE

The RHEV implementation has been deployed on the same hardware used for the first one and upgraded in terms of memory. The specifications about the memory and the I/O cards are summarised in Table 2.

Table 2: Hardware Specifications

CPU	2 x E5530 @ 2.4 GHz (8 real cores + Hyper Threading)
Memory	6 x 8 GB = 48 GB RAM
Network adapters	2 x 10 Gb network interfaces (for VLAN sharing, 1 linked to ECS) 2 X 1 Gb network interfaces (1 linked to CERN GPN, 1 used for cluster communications)
Fibre channel adapters	2 X 8 Gb Fiber channel switches (linked to two isolated fabrics)

STORAGE AREA NETWORK (SAN)

Virtual disks are stored on a DDN 9900 shared storage system as logical volumes (LV) interconnected through a redundant multipath fibre channel.

The DDN 9900 storage controllers can reach a high level of throughput using a proprietary RAID level called "Direct RAID": each RAID set consist of 10 spindles (disks), of which 8 are used for the data and 2 for the parity.

Three SATA RAID sets ("tiers") and one small SSD RAID set have been exported to the RHEV cluster and configured as a single Volume Group (VG).

The preferable block size for the Logical Units (LUN) in the RAID set is a multiple of 4 Kilobyte (512 Bytes times 8 disks), but unfortunately this value is not supported either by both Hyper-V nor by RHEV, which force us to use a LUN block size of 512 Bytes with the consequential lost of performance in terms of bandwidth and random Input Output Operation per Seconds (IOPS)

STORAGE IOPS

The reason why the DDN9900 is very fast in terms of throughput is the simultaneous access to all the disks in the same RAID set. While this makes the storage extraordinarily fast for sequential I/O operations, the access to all disks at the same time drastically reduces the number of random IOPS.

Considering that each SATA RAID set can perform ~200 random IOPS and that a virtual machine (VM) for standard operations needs at least an average of 30 IOPS, the current storage implementation limit the maximum number of VMs to ~50 using 4 RAID-sets. The effect of adding IOPS is clearly visible in Figure 3.

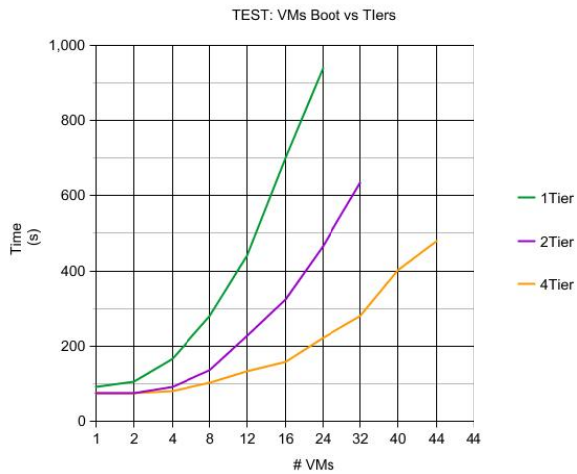


Figure 3: shows how increasing the number of RAID sets (IOPS) improves performances measured in "boot time".

TUNING

Even though there is a fundamental limitation in our current storage hardware a lot of improvement can be obtained by carefully tuning various parameters described in the following.

The main improvements, after having increased the number of RAID sets to the maximum available, have been achieved by switching the VMs default scheduler to NOOP:

By default Red Hat/Scientific Linux uses the CFQ [5] scheduler configured to balance the IO request and it aggregate them to a smallest number of large requests. While the idea of adding "intelligence" to the scheduling of the IO requests is great for a real PC, in a virtual machine this kind of scheduling will just add an additional delay since the smart scheduling will be done

twice: one time by the virtual machine and one time by the hypervisor in which the virtual machine is running.

Significant improvements have been measured as well adding to the default mount options of the VMs filesystems the "noatime,nodiratime" option as well as mounting the /tmp as tmpfs. This last measure keeps temporary files on a local RAM disk rather than soliciting external storage.

By default in ext3 for each read-request a write request will be triggered in order to update the file and directory access time.

Starting a "Name Server/LDAP" caching daemon and disabling IPv6 improved the users experience for the kind of virtual machines which are dedicated to general log-in. In this way the virtual machine does not need to do a request to the DNS and LDAP servers every time an hostname, an IP address or a UID/GID needs to be resolved.

For the virtual machines that are creating a lot of IOPS a solution based on moving the ext3 metadata away from the data in a dedicated SSD RAID set has been put in place. The details of this will be described in a forthcoming, dedicated publication.

This solution which is order of magnitude faster in terms of IOPS compared to the SATA RAID set works also for real machines and is achieved through LVM moving the physical extents in which the metadata is stored to an SSD RAID set.

Regarding the tuning of the hypervisors particular attention was given to the fiber channel interfaces in which we decreased the frame size to 512 Bytes allowing a more number of frames and consequentially of IOPS to the storage in the same interval time.

LHCB VIRTUAL NETWORKS

Live migration of VMs is one of the main advantages of having a virtual infrastructure making the machines less vulnerable to HW failures.

This put same constraint on the network configuration and according to common security procedures three virtual firewalls based have been put in place in order to isolate virtual networks and demilitarized zones. These are shared between the real machines using VLAN through a 10 Gb/s link.

The two 1 Gb/s links are dedicated respectively to cluster management communications and as up-link to CERN network/Internet.

For high-availability reasons the LHCb networks have been linked through a 10Gb/s connection per server with a switch uplink the LHCb core router of 20 Gb/s made by two link on two different linecards using the Link Aggregation Control Protocol (LACP).

A logical map of the virtual network is illustrated in Figure 4.

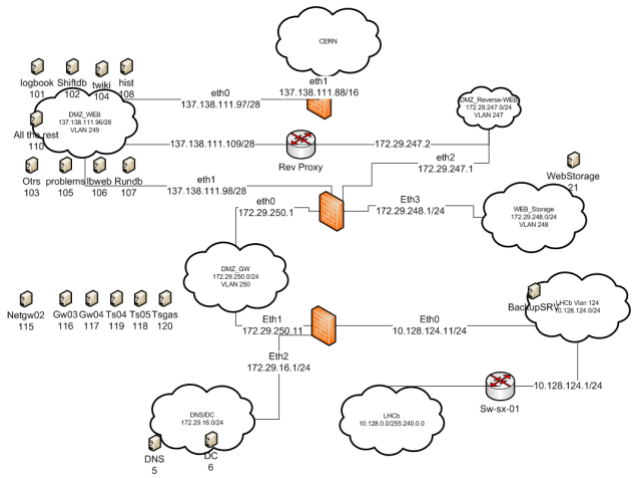


Figure 4: Logical Virtual Networks.

NETWORK PERFORMANCES

We measured the network throughput and the network latency from a KVM and a hyper-v virtual machine with the paravirtualized drivers installed, to a real server inside the LHCb network linked to the core router.

The tests have been done with iperf [6] and ICMP echo requests/replies.

The results are for KVM respectively ~1.50 Gb/s of throughput and ~0.3 ms of latency and for Hyper-V ~900 Mb/s of throughput and ~0.2 ms of latency.

In both hypervisors when the network traffic is filtered and routed by an additional virtual machine the latency time increases to ~0.6 ms and the bandwidth decrease to ~250 Mb/s

INTEGRATION WITH QUATTOR CLUSTER MANAGEMENT TOOL

The main problem in deploying OS on a Hyper-V virtual machine is the lack of pre execution environment (PXE) support when paravirtualized driver are used.

This is not the case for the KVM based VMs because the paravirtualized drivers called VIRTIO are included in the main vanilla kernel since version 2.6.20 and ported back by RedHat/Scientific Linux to version 2.6.18.

The virtual machines are now installed using QUATTOR, a system administration toolkit that provides a powerful, portable, and modular set of tools for the automated installation, configuration, and management of linux clusters and farms, like any real machine in the experiment [7].

ISSUES

Unlike the first study done on Microsoft Hyper-V we did not find any problems with networking, multicast and ACPI, also licensing problems with PVSS are not present. The main problem is the current storage backend, whose RAID system is not optimized for random IOPS.

CONCLUSIONS

The LHCb virtual infrastructure is now based on RHEV. Careful tuning allowed us to achieve very good latency and network and storage throughput.

The current system is however affected by a bottleneck in terms of random IOPS limiting the number of VMs to be executed at the same time.

A new storage solution will be chosen and bought in Q4 2011. The requirements derived from studies conducted for this paper will guarantee a maximum number of concurrent VMs of at least 180.

Once the acquisition will be completed we will continue to deploy more VMs focusing on the control PCs of the experiment.

REFERENCES

[1] LHCb Trigger System TDR, LHCb TDR 10, CERN/LHCC/2003-31, 2003.

[2] An Integrated Experiment Control System, Architecture, and Benefits: The LHCb Approach – IEE Transaction on Nuclear Science, Vol. 51, NO 3, June 2004.

[3] The LHCb Trigger and Data Acquisition System J.-P. Dufey, M. Frank, F. Harris, J. Harvey, B. Jost, P. Mato, H. Mueller.

[4] Virtualization for the LHCb online system – CHEP 2010 – ID 141. E. Bonaccorsi, L. Brarda, G. Moine, N. Neufeld.

[5] http://www.redhat.com/f/pdf/rhel/Oracle-10-g-recommendations-v1_2.pdf.

[6] http://iperf.sourceforge.net/.

[7] https://lbtwiki.cern.ch/bin/view/Online/AdminGuide Quattor.

NETWORK SECURITY SYSTEM AND METHOD FOR RIBF CONTROL SYSTEM

A. Uchiyama#, R. Koyama, SHI Accelerator Service Ltd., Shinagawa, Tokyo, Japan
M. Komiyama, M. Fujimaki, N. Fukunishi, RIKEN Nishina Center, Wako, Saitama, Japan

Abstract

A closed network is more reliable for accelerator control from the viewpoint of information security. The control system of RIKEN RI Beam Factory (RIBF) based on EPICS is also constructed on a closed network that is completely disconnected from all the other networks including laboratory intranets used for daily research activities. However, there are several inconveniences in exchanging information between the network of RIBF control and others. To solve the conflicts, we designed and implemented the network security system. This proceeding describes the design policy of the system and the method used for the exchange information between the intranet in RIKEN and the closed network used for RIBF control system.

INTRODUCTION

The control system of RIKEN RIBF adopted the EPICS (Experimental Physics and Industrial Control System) that used LAN-based protocols [1] and was constructed on its own network (ACC-LAN). E-mail services and Internet accesses unrelated to accelerator operations utilize the RIKEN virtual LAN (RIKEN-VLAN) which is a major network system in RIKEN Wako campus. In the RIKEN-VLAN, Wireless Fidelity (Wi-Fi) routers have been installed in all of the buildings, and all visitors including beam users can access the Internet using the Wi-Fi with DHCP services in RIKEN.

ACC-LAN, however, should not be connected directly to the RIKEN-VLAN because there are possibilities of illegal accesses of intruders to ACC-LAN via Wi-Fi services of the RIKEN-VLAN, even if illegal accesses from wide area network (WAN) is denied with a firewall.

Although the use of the network disconnected from RIKEN-VLAN and WAN is effective for network security of RIBF control system, there were several inconveniences. In this network system, it is difficult for members of the accelerator group to monitor the status of accelerator operation in real time from their offices because access ports of ACC-LAN are not prepared for the offices of the members. The members were hence required to use a storage device, such as USB flash memory, to extract logged data of accelerator operation from ACC-LAN. It was also impossible to transfer E-mail alerts, which are widely used for server management. In addition, RIBF beam users frequently request wide varieties of information, such as accelerator operation status. For these reasons, we decided to improve the inconvenient situation without risking information security.

NETWORK ARCHITECTURE

The ACC-LAN consists of Ethernet switches, optical fibres and metal cables, which are all commercially available. The control system comprises two different systems: an accelerator control system based on EPICS and the other is a system for non-EPICS-based utility control (see Fig. 1). The controllers, servers, and client PCs are installed on the EPICS-based network system. On the other hand, some digital measurement instruments, video servers, and network cameras are connected to the network for non-EPICS-based utility control.

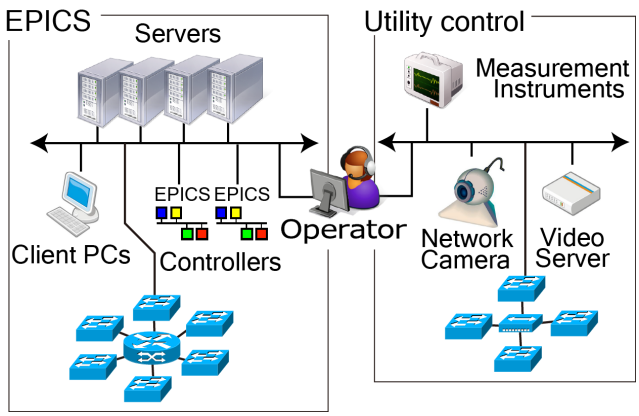


Figure 1: Outline of the ACC-LAN.

PROTECTION AGAINST MALWARE

Nowadays, spread of malware, such as warm, becomes one of the serious issues on the Internet. For example, according to Adobe.com report, systems using Microsoft Windows, Macintosh, Linux and Solaris are possibly infected by malware due to critical vulnerability in old versions of Adobe Flash Player, Reader and Acrobat [2]. Usually the accelerator operation does not require Internet connections and E-mail services. Since major parts of malware infections are caused by Web browsers and E-mails, communications to the Internet or RIKEN-VLAN from client PCs used in accelerator operation are strictly prohibited.

#a-uchi@riken.jp

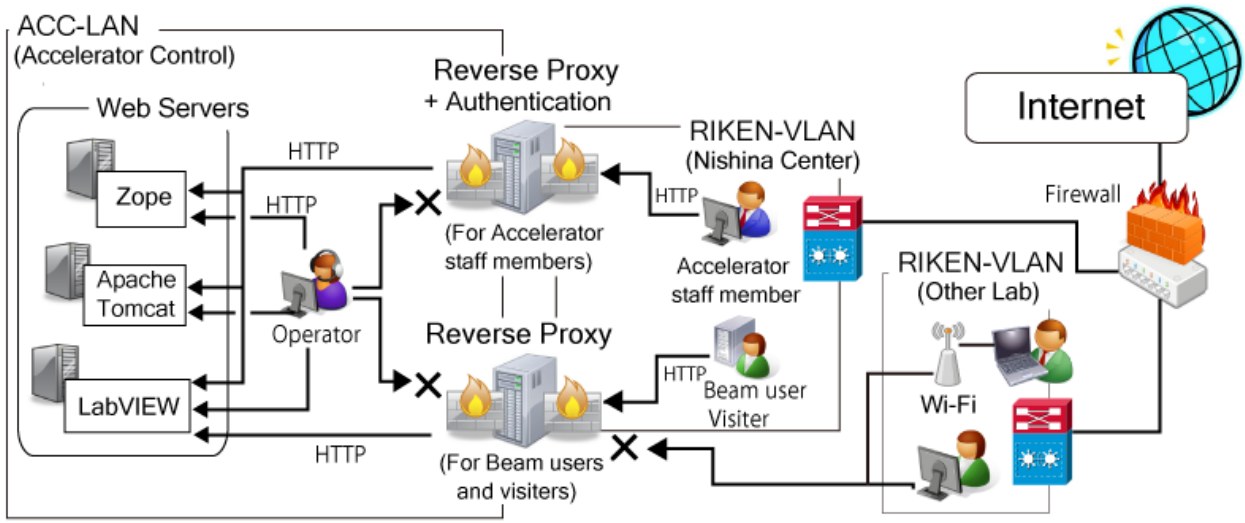


Figure 2: Network diagram in the web communication using reverse proxy servers between ACC-LAN used for RIBF control system and RIKEN-VLAN for office network.

WEB COMMUNICATION USING REVERSE PROXY

System Construction

From the view points of the system management and easiness of system construction, web communication is a suitable to provide information stored in ACC-LAN to many people. To meet the requirements, a combined system with reverse proxy servers for web communication and a firewall has been constructed for providing accelerator information to RIKEN-VLAN with ensuring secured access. Installation of reverse proxy servers, in front of real web servers, is widely used prescription for security and caching.

In order to implement the system, the standard package of CentOS 5.5 has been used as the operating system. Further, Iptables [3] for the firewall and Squid [4] for the reverse proxy server are installed. The system chart and basic concepts are shown in Fig. 2. The significant feature of our system is that accesses from RIKEN-VLAN are masked by the web servers behind the reverse proxy servers. As a result, our system is protected from attacks.

Access Control of Information

In order to restrict unnecessary accesses from beam users and visitors to critical information of accelerator control, such as operation log and raw data, we have adopted an authentication system. In order to control the access of users by authentication, our system consists of two reverse proxy servers. To manage the username/password and denial or allowance to hosts from each user easily, accelerator staff members and beam users/visitors use different reverse proxy server. Therefore, the environments that allow users to access web sites are different for accelerator staff members and beam users/visitors, because the content of information is different. Beam users/visitors are permitted to access only some specific website without the authentication, while accelerator staff members can access all the registered web servers in ACC-LAN with authentication. Consequently, single sign-on (SSO) authentication has been achieved by choosing the accessible reverse proxy server for the accelerator staff members, because SSO has a mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems.

Protection against Illegal Access

In order to prevent illegal accesses using SSH and other protocols, the revers proxy system opens only a minimum number of ports to the reverse proxy servers by installing firewall. Additionally, access control based on network addresses was implemented, that is accesses using HTTP protocol required from the PCs having Nishina Center's network addresses in RIKEN-VLAN. Furthermore, the authentication system is used not only for the access control but also play an essential role for the protection against illegal access. Actually, it is difficult to test vulnerabilities against all the type of attacks, for example cross site scripting, for all of the web sites inside ACC-LAN. However in this case, the administrator needs to test only a small number of web sites which visitors can access without authentication.

Result of Access Test

The following software and devices, which support a web interface or the HTTP protocol, have been used inside ACC-LAN.

- Zlog (Zope based Operational Log System) [5]
- MyDAQ2 (MySQL based Archive system) [6]

- Wiki-based log system for 28GHz Ion Source [7]
- Wiki-based documents for RIBF control system
- RIBFCAS (RIBF control data archive system) [8]
- Video server (produced by AXIS using ActiveX)
- Network camera (produced by Sony using ActiveX)
- Digital oscilloscope (produced by Agilent)
- Digital measurement instrument (MW100 produced by Yokogawa Electric Corporation)

We executed access tests to check if the ones listed above in ACC-LAN can be accessed from RIKEN-VLAN via the reverse proxy servers. As a result, it was confirmed that all of them except the digital oscilloscopes and MW100, which have web interface using Java applet, can be accessed successfully.

Other Web Services

The present method enables us to develop a new type of software of RIBF control system with HTTP protocol because process variables (PVs) of EPICS can be accessed from RIKEN-VLAN via reverse proxy server. PVs are converted into XML and JSON formats by our frame-work, which run on the apache and is written in PHP by using PHP EPICS module developed by PSI [9]. Since many application programming interfaces, such as web application have been adopted XML and JSON format recently, it has scalability for the system development. For this reason, using PVs converted XML and JSON formats, we can develop EPICS clients running on RIKEN-VLAN without the use of EPICS channel access protocol.

Additionally, some applications have been provided by using LabVIEW's HTTP service after construction of this revers proxy system. One example of these applications is "Beam Transport Map" developed by R. Koyama et al. [10]. This web application provides real-time operation status of RIBF accelerators with members, users of RIKEN Nishina Center via this reverse proxy. Consequently, various types of monitoring data and status in ACC-LAN become commonly communicated in RIKEN-VLAN by using HTTP protocol (See Fig. 3).

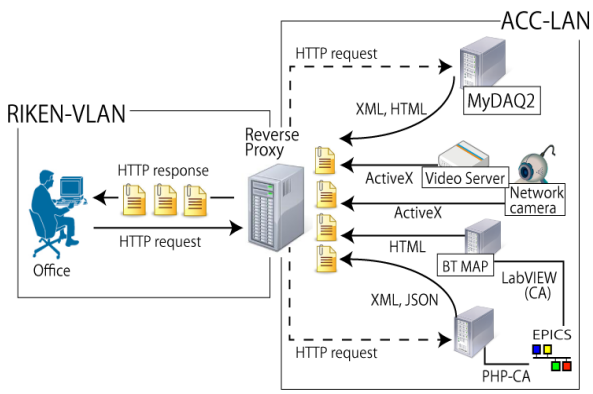


Figure 3: Integrated communication with HTTP protocol.

SOFTWARE UPDATE

Generally, almost all Linux distributions are updated by using YUM and APT. However, it is not easy to update software in a closed network, when it is required to fix security bugs. In the RIBF control system, almost all servers and clients have adopted CentOS and Scientific Linux based on Redhat Enterprise Linux. Therefore, we constructed our own YUM server inside the ACC-LAN. When a serious security hole will be reported, we will install an updated package in our YUM server as soon as possible.

E-MAIL ALERT

Although, it was successfully proceeding to implement the reverse proxy system in RIBF control system. The present work is not completed and the introduced system is not a perfect risk-free system from the view point of information security. As an example of perfect risk-free system in sending information from a completely closed network system, we introduced a new E-mail alert system. It is also important to notify administrators via E-mail automatically on serious failures for management of servers and systems. Since RIBF control system did not have this feature, we designed and implemented a system to send E-mail alerts from EPICS. The present alert system sends E-mail by using on-off control action of EPICS-based mechanical switches without a connection between the networks.

The system consists of an E-mail auto-sender embedded board (mailer board, See Fig. 4) manufactured by TriState ltd. [11], Yokogawa FA-M3 as Programmable Logic Controller (PLC) and Linux Input/Output Controller (IOC). The system chart is shown in Fig. 5. The main characteristic of the mailer board is sending E-mail to a specified mail address that is determined by 16 DIs. The mailer board is installed as a standalone E-mail client in the RIKEN-VLAN. The PLC is installed as one of EPICS control devices within ACC-LAN. As a result, the E-mail alert from EPICS IOC installed in ACC-LAN was sent to RIKEN-VLAN with secured access, though the each network is separated completely. In RIBF control system, it has been used as E-mail alert to administrators in case of system troubles, such as interruption of important services, and reboot of EPICS IOCs.

We have succeeded in sending information using hardware signal in a perfectly closed network. Currently, the reverse proxy system and E-mail alert system are being weighed from the view point of ensuring security.

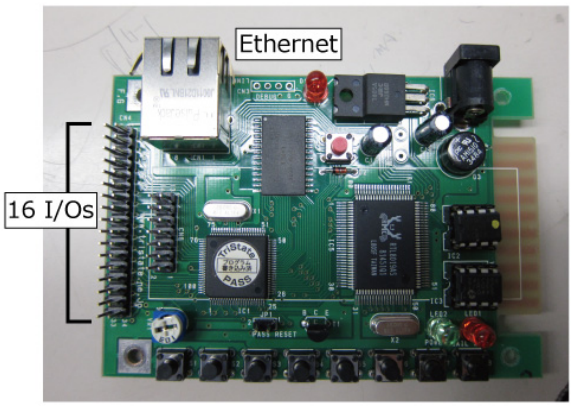


Figure 4: Photograph of the E-mail auto-sender embedded board manufactured by TriState Ltd.

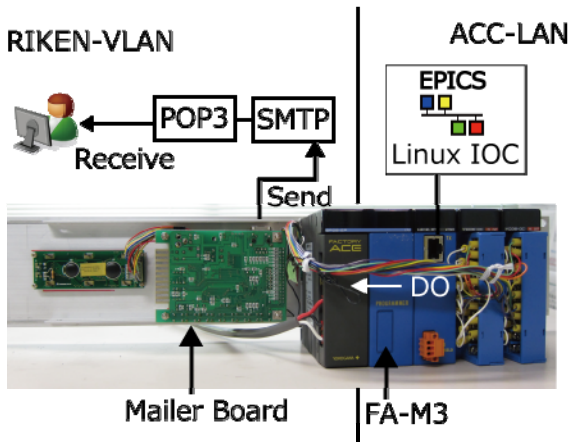


Figure 5: The system chart of E-mail alert.

SUMMARY

Reverse proxy servers, which interconnect the RIBF control network and the RIKEN-VLAN for providing various types of services in the RIKEN Wako campus, were designed and constructed. The HTTP protocol was adopted for the reverse proxy servers to allow accelerator staff members, beam users and other staff members to access the real-time information on operational status of accelerators with ensuring security against possible attacks via the RIKEN-VLAN. In addition, an E-mail alert system has been constructed by using an E-mail auto-sender board on the RIKEN-VLAN and a PLC-based IOC, prepared in ACC-LAN. The E-mail alert system helps the administrators to notice quickly when serious problems occur. It was confirmed that reverse proxy system is an effective method to provide information ensuring secured access in RIKEN RIBF. The usefulness of E-mail alert system without network connection was also proved by operational experience.

ACKNOWLEDGEMENT

One of the authors (A.U) would thank to Prof. K. Furukawa of KEK for useful discussion and valuable comments.

REFERENCES

[1] M. Komiyama et al., "Status of Control System for RIKEN RI-Beam Factory" Proc. ICALEPCS07, Knoxville, Tennessee, USA, 2007, p.334
[2] http://adobe.com
[3] http://www.netfilter.org/projects/iptables/
[4] http://www.squid-cache.org/
[5] K. Yoshii et al., "Web-Based Electronic Operation Log System – Zlog System" Proc. ICALEPCS07, Knoxville, Tennessee, USA, (2007), p.299.
[6] T. Hirono et al., "Development of Data Logging and Display System, MyDAQ2" Proc. PCaPAC08, Ljubljana, Slovenia, (2008), p. 55.
[7] A. Uchiyama et al., "Construction of Client System for 28GHz SC-ECRIS" RIKEN Accel. Prog. Rep. 43, p. 133, 2009.
[8] M. Komiyama et al., in these proceedings.
[9] A. Bertrand et al., "EPICS on the WEB" Proc. 10th ICALEPCS, Geneva, Swiss, (2005), P3_087.
[10] R. Koyama et al., "Development of "BTmap": Online visualization of beam-transport status" RIKEN Accel. Prog. Rep. 44 (Accepted).
[11] http://www.tristate.ne.jp/mailer02.htm

VIRTUAL IO CONTROLLERS AT J-PARC MR USING XEN

N. Kamikubota, S. Yamada and N. Yamamoto, KEK / J-PARC, Tokai-mura, Ibaraki, Japan
T. Iitsuka, S. Motohashi, M. Takagi, S. Yoshida
Kanto Information Service (KIS), Tsuchiura, Ibaraki, Japan
H. Nemoto, ACMOS Inc., Tokai-mura, Ibaraki, Japan

Abstract

The control system for J-PARC accelerator complex has been developed based on the EPICS toolkit. About 90 traditional ("real") VME-bus computers are used as EPICS IOC's in the control system for J-PARC MR (Main Ring).

In 2010-2011, we demonstrated a "virtual" IOC using Xen, an open-source virtual machine monitor. Scientific Linux with an EPICS IOC runs on a Xen virtual machine. EPICS records for oscilloscopes (network devices) are configured. Advantages of virtual IOC are discussed.

In addition, future directions are discussed. Plan view of virtual IOC's for MR operation will be given.

INTRODUCTION

The J-PARC Main Ring (hereafter MR) is a high-power proton synchrotron with beam-energy 30GeV. The beam commissioning of MR started May, 2008. Since then, various machine studies as well as beam deliveries to physics experiments have been carried out [1]. The control system for MR has been developed based on EPICS (Experimental Physics and Industrial Control System) [2], where EPICS is a toolkit for large accelerator controls developed and supported by an international community [3].

Table 1: Types of IOC used in J-PARC MR

Type	Number	Comment
VME-bus GE and Sanritz	~90	Traditional styles
Yokogaw PLC F3RP61-2L	~30	Embedded EPICS [4] with PLC I/O modules
Cosylab microIOC	3	Industrial PC [5] with serial lines (RS485)

In EPICS, signals are handled by an in-out controller (hereafter IOC). Traditional EPICS systems use a VME-bus computer as an IOC. In MR, we have about 90 VME-bus-type IOC's among 120 IOC's in total (Table 1). A snapshot of IOC status overview during operation is given in Fig. 1.

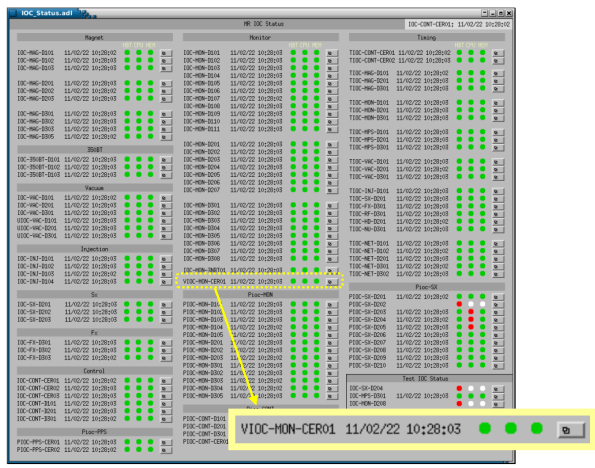


Figure 1: Overview for J-PARC MR IO-Controllers.

In J-PARC MR, 60 of 90 VME-bus-type IOC's do not use VME-bus. They are used for network-based devices: such as PLC with ladder sequences, intelligent measuring systems (WE7000 [6]), oscilloscopes, etc [7,2]. Typical topology found in MR is shown in Fig. 2.

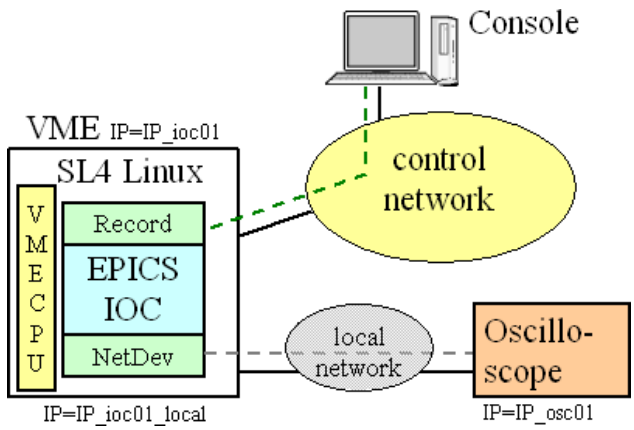


Figure 2: Typical topology with a network device.

STUDY OF VIRTUAL IOC

Idea of Virtual IOC

Recent IT technology has provided various types of virtualization of a computer: for example, Xen, VMware, VirtualBox, and KVM. Among them, we firstly interested in Xen [8], an open-source virtualization technology. The start procedures of a "vioc - virtual IOC" is given below (see also Fig. 3):

(1) Prepare a host OS. In our case, it is a blade-type server running Scientific Linux 5 (SL5).

(2) Run a guest OS using Xen. The guest OS is assigned 512MB memory, running SL5, with an independent IP address from the host.
(3) Start an EPICS IOC on the guest OS. The EPICS IOC's in Fig. 2 and Fig. 3 are identical.

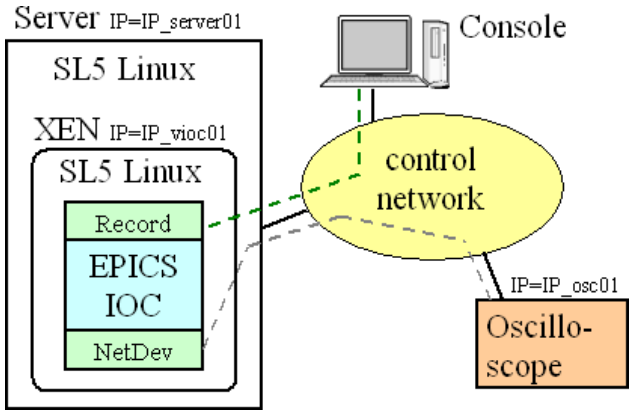


Figure 3: Topology with a virtual IOC.

Demonstration of VIOC at J-PARC MR

In order to demonstrate a vioc at J-PARC MR, we selected an extinction-monitor setup [9]. It consisted of oscilloscopes, and not EPICS-based. Thus, we developed an EPICS-based system using vioc in October, 2010. The system controls Tektronix DPO4054 and TDS3054 oscilloscopes. Setting parameters as well as waveform signals (four input and 1 output) are configured as EPICS records. Typical waveform is 10,000 elements, 10 us with 1G/s sampling rate.

The vioc for extinction monitor setup started operation since November, 2010. It was used in machine studies very successfully. Moreover, the vioc was included in our IOC surveillance monitor with the name "VIOC-MON-CER01" (Fig. 1). It had been stable over a few months. It is worth noting that nobody can find it a "virtual", not a "real" VME-bus-type, unless using it remotely (Fig. 4).

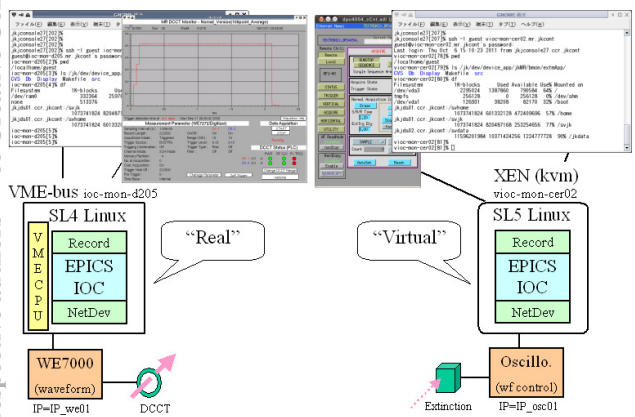


Figure 4: virtual IOC (right) and real IOC (left).

Discussion

Based on our successful experience, a scheme that running multiple vioc's (guest OS's) on limited number of servers (host OS's) seems promising. It is efficient and flexible: (a) adding new vioc is easy, (b) customization for each vioc (i.e. small memory size for small function) is possible. When a server has a trouble, one can move vioc's to another server.

However, we must pay attention to network traffic. Unless network amount is small, vioc scheme works well. When waveform signals are handled at a considerable high repetition rate, "real" scheme has an advantage. More studies are needed.

FUTURE DIRECTIONS

Xen to KVM

We have demonstrated a vioc using Xen. However, recent trend of virtualization technology suggests KVM. KVM functions were merged into Linux kernel after 2.6.20. Now Scientific Linux 6 (SL6) has native supports for KVM.

When we run multiple vioc's on servers, it is apparent that we need a management tool. In the early phase, we looked for a tool for Xen, but could not find an appropriate one. For KVM, a default "virt-manager" is available with SL6. We have checked that it is good and stable enough for our purposes.

Toward mass introduction of vioc's in the future, we decided to use KVM instead of Xen. A plan view of vioc's, using KVM and SL6, is shown in Fig. 5. We will start operation with the new scheme after December, 2011.

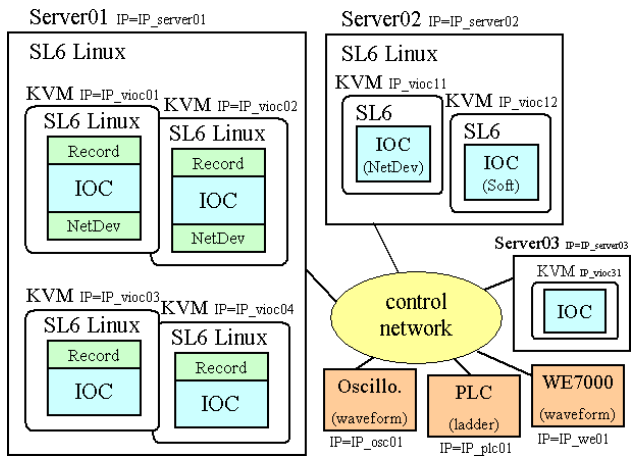


Figure 5: Plan view of vioc's for future MR operation.

Virtual Terminal

We also studied a possibility to use virtual desktop infrastructure (VDI) under KVM environment. The idea of "vterm - virtual terminal" is similar to vioc. Instead of an EPICS IOC, we run a SPICE server on a virtual machine. SPICE is a remote-desktop protocol supported by recent Linux kernels [10]. On a SPICE server, we run GUI applications developed in our EPICS-based control system.

We have an operator's console system using commercial thin-client terminals [11]. We expect higher reliability when we run a SPICE client on a thin-client terminal, since system load (CPU and memory) is always constant. Number of GUI applications does not affect to

stability of a thin-client terminal. Studies are under progress.

CONCLUTION

We have demonstrated a virtual IOC (vioc) on a Xen virtual machine. A vioc was configured for parameter settings of oscilloscopes. It has been used very successfully in studies of extinction monitors. In addition, it was stable over a few months.

Recently we decided to usie KVM virtualization technology instead of Xen. After December, 2011, we will start to run multiple vioc's in our operation.

REFERENCES

[1] T.Koseki et al, "Challenges and Solutions for J-PARC Commissioning and Early Operation", IPAC'10, Kyoto, Japan, p.1304-1308
T.Koseki et al, "Present Status of J-PARC MR Synchrotron", IPAC'10, Kyoto, Japan, p.259-261
T.Koseki et al., this conference
[2] H.Yoshikawa et al, "Current Status of the Control System for J-PARC Accelerator Complex", ICALEPCS 2007, Knoxville, P.62-64
N.Kamikubota, "J-PARC Status", presentation in EPICS Collaboration Meeting in Shanghai, Mar. 2008
N.Kamikubota, "J-PARC Control toward Future Reliable Operation", this conference
[3] http://www.aps.anl.gov/epics/
[4] J.Odagiri et al, "Application of EPICS on F3RP61 to Accelerator Control", ICALEPCS2009, Kobe, Japan, p.916-918
A.Uchiyama et al, "Development of Embedded EPICS on F3RP61-2L", PCaPAC2008, Ljubljana, Slovenia, p.145-147
[5] http://www.microioc.com/
[6] M.Takagi, "Network-based Waveform Monitor for J-PARC Accelerator Complex", ICALEPCS 2003, Gyeongju, Korea, p.497-499
[7] J.Odagiri et al, "EPICS Device/Driver Support Modules for Network-based Intelligent Controllers", ICALEPCS 2003, Gyeongju, Korea, p.494-496.
N.Kamikubota, "Applications of Network-based Controllers at KEK", presentation in EPICS Seminar in Shanghai, April 2005
[8] http://xen.org
[9] K.Yoshimura et al, "Measurements of Proton Beam Extinction at J-PARC", IPAC'10, Kyoto, Japan, p.984-986
[10] http://www.spice-space.oeg
[11] S.Yoshida et al, "Console System using Thin Client for the J-PARC Accelerators", ICALEPCS 2007, Knoxville, P.383-385

PACKAGING OF CONTROL SYSTEM SOFTWARE

K. Zagar, M. Kobal, N. Saje, A. Zagar, Cosylab, Ljubljana, Slovenia,
R. Sabjan, COBIK, Solkan, Slovenia,
F. Di Maio, D. Stepanov, ITER Organization, St. Paul lez Durance, France

Abstract

Control system software consists of several parts – the core of the control system, drivers for integration of devices, configuration for user interfaces, alarm system, etc. Once the software is developed and configured, it must be installed to computers where it runs. Usually, it is installed on an operating system whose services it needs, and also in some cases dynamically links with the libraries it provides. Operating system can be quite complex itself – for example, a typical Linux distribution consists of several thousand packages. To manage this complexity, we have decided to rely on Red Hat Package Management system (RPM) to package control system software, and also ensure it is properly installed (i.e., that dependencies are also installed, and that scripts are run after installation if any additional actions need to be performed). As dozens of RPM packages need to be prepared, we are reducing the amount of effort and improving consistency between packages through a Maven-based infrastructure that assists in packaging (e.g., automated generation of RPM SPEC files, including automated identification of dependencies). So far, we have used it to package EPICS, Control System Studio (CSS) and several device drivers. We perform extensive testing on Red Hat Enterprise Linux 5.5, but we have also verified that packaging works on CentOS and Scientific Linux. In this article, we describe in greater detail the systematic system of packaging we are using, and its particular application for the ITER CODAC Core System.

INTRODUCTION

The principal challenges of today's control systems for large experimental physics facilities are complexity and quality assurance.

By this we mean the fact that a large number of software components – executing either on the same host or in a distributed set-up – need to be integrated into a functioning whole while performing according to performance and reliability expectations. The complexity challenge stems both from the inherently distributed, large-scale nature of a control system, as well as the trends in component-based software engineering and systems engineering, where monolithic systems are giving way for those that are integrated from smaller, more manageable subsystems.

With limited development, maintenance and integration resources – in particular skilled staff – it is important that as many tasks as possible are automated, and that common problems have common solutions – i.e., that standardization takes place to the extent that it is economically feasible.

The ITER CODAC [1] control system is also facing these challenges. CODAC integrates software packages that are a product of two decades of work, and which have been developed in diverse environments by different teams. Not surprisingly, each of these packages takes a different approach on how the software is built, and what quality assurance process is in place during its release.

We have decided to standardize at least the interface with which the developer (or maintainer) interacts with the build system. To achieve this, we have wrapped the diverse approaches and technologies for building (Makefile [2], Ant [3], shell scripts, Eclipse builder [4], etc.) into one tool. Since many software packages share the same approach (e.g., EPICS base [5] and all of its extensions rely on Makefile, while Control System Studio [6] and all of its plug-ins rely on Eclipse), we were looking for a way to re-use our effort: for example, specify integration with the EPICS Makefile system in a single place, and "invoke" it with a one-line stanza in all software packages where it is needed.

As we are not the first to have come across this challenge, a market survey revealed that build tool frameworks already exist (for example, Maven [7] and Gradle [8]). After our evaluation, performed in late 2009, we have settled to use Maven 2 as the platform, and we have chosen to solve our challenges by implementing a plug-in for this tool.

Another challenge is managing deployment across the many hosts that will eventually constitute the control system. However, this challenge is not uncommon in the IT industry, where large corporations also have thousands of computers that need to be managed with the limited IT staff. To leverage existing solutions, ITER had decided to take an off-the-shelf approach: using Red Hat Enterprise Linux and its automated installation and update capabilities enabled with the Red Hat Satellite software [9].

Managing installation, un-installation and updating of software packages on an individual host is a rather complex task in itself. The most trivial step of it is to place the files constituting a software package (executables, scripts, configuration files and data files) to the right places in the file system. As un-installation and update need to be able to clean-up those files, meta-data must be associated with each file to specify which software package had installed it. Installation/un-installation might require that some actions are taken (e.g., adaptation of configuration files of other software components, creation of database schemas, population of databases, etc.). And finally, software package might have dependencies, and other software might depend on: installing a package might thus have a precondition that

```
# Spec file for package rf-ich-sample-MCioc
# Generated by the codac-packager Maven plugin.
# Date: Fri Sep 30 13:32:55 CEST 2011 ...

%define unit_version_full %{codac_version_full}.v1.0a1

Name:    %{codac_rpm_prefix}-rf-ich-sample-MCioc
Version: %{codac_version_full}.v1.0a1
...

Requires:        %get_current codac-core-3.0-epics-autosave

%description
Input (a snippet from Maven pom.xml):

%install
sed -r -i 's#epicsEnvSet\("TOP"\s*\,\s*".*?"\)#epicsEnvSet\("TOP","/opt/codac-3.0/apps/rf-ich-sample"\)#g'
%{buildroot}/opt/codac-3.0/apps/rf-ich-sample/iocBoot/iocMC-ICHCore/envPaths
...

install -d %{buildroot}/etc/opt/codac-3.0/alt.d/
echo --slave \"%{_bindir}/MC-ICHCore-ioc\" \"codac-sudo-MC-ICHCore-ioc\" \"/opt/codac-3.0/bin/services/sudo-
service\" >> "%{buildroot}/etc/opt/codac-3.0/alt.d/rf-ich-sample-MCioc"
echo --slave \"%{_initrddir}/MC-ICHCore-ioc\" \"codac-srv-MC-ICHCore-ioc\" \"/opt/codac-3.0/bin/services/MC-
ICHCore-ioc\" >> "%{buildroot}/etc/opt/co

%pre
...
```

Figure 1: Example of a SPEC file that provides meta-information and build instructions for an RPM package.

other packages are installed beforehand, and uninstalling it may have a consequence that those depending on it should be uninstalled as well.

This problem, too, has already been solved by the IT community. On Linux platforms, the Debian package management (APT/DEB) and Red Hat Package management (YUM/RPM) are commonplace. As ITER has chosen Red Hat Enterprise Linux as the operating system, we have opted for the YUM/RPM technology.

To provide a RPM, the developer must provide a so-called SPEC file. The SPEC file contains (see Figure 1):

- meta-information about the package (name, version, description, etc.),
- instructions in form of an executable script on how to build the package (unpack the sources, run configure/make or other tools, etc.)
- which files to package, and what default permissions to assign to them
- the scripts to execute before and after installation, and before and after un-installation.

CONSTITUENTS OF A CONTROL SYSTEM

In case of ITER CODAC, the control system consists of the following kinds of software packages:

- EPICS IOC applications.
- Configuration files for the BEAST alarm server.
- Configuration files for the BEAUTY archiving system.
- Kernel modules, e.g., for implementation of kernel-mode device drivers.
- User-mode device drivers and libraries.

The EPICS IOC applications are standard EPICS applications, built with the EPICS' Makefile system. They consist of the binary compiled for the target platform, the `st.cmd` start-up script, and EPICS database files.

These files are packaged in an RPM package, and an `init.d` script is automatically generated that allows for starting up and shutting down of the IOC as a system service. The `init.d` script also provides a console through which developers and maintainers can access the EPICS shell of the IOC process (via the `screen` tool).

For security reasons, it is not advisable to run services as the `root` user. Therefore, a system user called "`codac`" is provided, and all services run under that account. This raises some issues with permissions (e.g., the `codac` user by default doesn't have permissions to interact with kernel-mode device drivers, nor does it have permissions to set its real-time attributes such as scheduling priority and CPU affinity). The packaging ensures also that these permissions issues are properly addressed.

Packaging of configuration files for BEAST and BEAUTY involves putting the configuration files in the RPM package, and running the database import tools upon installation of the RPM to populate the BEAST and BEAUTY configuration databases with their content. In CODAC, the content of these configuration files is automatically generated by the SDD tools [1], thus they are in-sync with the contents of the EPICS configuration database.

Kernel modules are built with standard tools for kernel modules. Currently, kernel modules can be built for two targets:

- A regular kernel.
- A real-time kernel.

```
<package name="MCioc">
  <include name="MC-ICHCore" type="ioc"/>
  <include type="boy" file="*"/>
  <include type="databrowser" file="power.plt"/>
</package>
```

Figure 2: Excerpt from Maven's POM XML file responsible for generating RPM SPEC file from Figure 1.

For each build, a separate RPM package is provided, which then has its dependency set as required (either to the `kernel` or to the `kernel-rt` package).

THE MAVEN PLUGIN

The packaging of control system's constituents described in the previous section into RPMs is performed by the a Maven plugin we have developed.

The Maven plugin is designed to execute the packaging task during the "`package`" phase of its lifecycle – i.e., after compilation and testing, but before installation and deployment.

The description of the packaging is very concise. Figure 2 shows an example of the Maven configuration that results in RPM SPEC file shown in Figure 1.

The RPMs thus produced allow installation of several versions of the ITER CODAC system simultaneously. E.g., version 2.1 and 3.0 can be installed at the same time, with the first residing in `/opt/codac-2.1` and the latter in `/opt/codac-3.0`. The developer can then switch between the two versions with a command `codac-version`, which redirects all the softlinks from the system's paths accordingly (via the *alternatives* system) and reconfigures environment variables.

In addition, the Maven plugin provides command-line options that facilitate generation of Maven's project definition file, the `pom.xml`. Thus, even developers not familiar with syntax of this file can configure their projects to be packaged.

USAGE EXAMPLE

The developer might use the Maven-based tools as follows.

Firstly, the developer would create a *project* (a *unit* in ITER's CODAC terminology) by executing:

```
mvn iter:newunit -Dunit=my-unit
```

This results in a subdirectory *m-my-unit* (the *m-* prefix is prepended due to ITER CODAC's naming convention for units). The directory structure conformant to CODAC's standards, and Maven's `pom.xml` file, are also created by this command.

The following sequence of commands creates an EPICS application, and then configures an IOC process to run that application. The type of the application here is "psh", referring to ITER CODAC's *Plant System Host*:

```
cd m-my-unit
mvn iter:newapp \
    -Dapp=PlantSystemHost \
    -Dtype=psh
mvn iter:newioc \
    -Dioc=PlantSystemHost \
    -Dapp=PlantSystemHost \
    -Dtype=psh
```

Now, the RPM can already be prepared by executing:

```
mvn package
```

It is also possible to conveniently start the resulting IOC:

```
mvn iter:run
```

CONCLUSION

The Maven-based tools that have been developed to facilitate the development process greatly simplify development of software for the control system – from input/output controller processes to extensions of the operator's graphical user interfaces.

The interface provided to developers is simplified so that even those who had no prior exposure to EPICS, Maven or RPM are able to create new EPICS-based, CODAC-compliant projects, build them, and ensure that results are packaged in an installable RPM.

While the tools have been developed for ITER, they can be adapted to other projects as well. For example, currently an effort to adapt ITER CODAC for the needs of the European Spallation Source's control system.

REFERENCES

[1] F. Di Maio et al., "The CODAC Software Distribution for the ITER Plant Systems", ICALEPCS'11, Grenoble, France.
[2] GNU Make; http://www.gnu.org/s/make/.
[3] Apache Ant; http://ant.apache.org/.
[4] Eclipse; http://www.eclipse.org/.
[5] Experimental Physics and Industrial Control System; http://www.aps.anl.gov/epics/.
[6] Control System Studio; http://cs-studio.sourceforge.net/.
[7] Apache Maven; http://maven.apache.org/.
[8] Gradle; http://www.gradle.org/.
[9] Red Hat: Red Hat Network Satellite; http://www.redhat.com/red_hat_network/.

TAILORING THE HARDWARE TO YOUR CONTROL SYSTEM*

E. Björklund, S.A. Baily, Los Alamos National Laboratory, Los Alamos, NM 87545, U.S.A.

Abstract

In the very early days of computerized accelerator control systems the entire control system, from the operator interface to the front-end data acquisition hardware, was custom designed and built for that one machine. This was expensive, but the resulting product was a control system seamlessly integrated (mostly) with the machine it was to control. Later, the advent of standardized bus systems such as CAMAC, VME, and CANBUS, made it practical and attractive to purchase commercially available data acquisition and control hardware. This greatly simplified the design but required that the control system be tailored to accommodate the features and eccentricities of the available hardware. Today we have standardized control systems (Tango, EPICS, DOOCS) using commercial hardware on standardized busses. With the advent of FPGA technology and programmable automation controllers (PACs & PLCs) it now becomes possible to tailor commercial hardware to the needs of a standardized control system and the target machine.

In this paper, we will discuss our experiences with tailoring a commercial industrial I/O system to meet the needs of the EPICS control system and the LANSCE accelerator. We took the National Instruments Compact RIO platform, embedded an EPICS IOC in its processor, and used its FPGA backplane to create a "standardized" industrial I/O system (analog in/out, binary in/out, counters, and stepper motors) that meets the specific needs of the LANSCE accelerator.

BACKGROUND

The 800 MeV proton accelerator at the Los Alamos Neutron Science Center (LANSCE) was designed and built in the 1960's. The original design included a custom-built computer control system based on a custom-built data acquisition system that we called RICE ("Remote Information and Control Equipment") [1]. In the 1980's, with the advent of the CAMAC standard, we adapted our control system to use both CAMAC and RICE. Then, in the 1990's, we introduced both VME and EPICS into our control system. This required a lot of adaptation – old control systems to new hardware, new control systems to old hardware, and new and old control systems to each other.

While it is nice to keep up with new technology, unfortunately it did not mean that we got to eliminate any of the old technology. And so, in the 2000's we found ourselves supporting a control system that included three generations of hardware technology (RICE, CAMAC, and VME) and two generations of software technology (EPICS and the legacy control system).

Now, at last, we have the opportunity to start phasing out our old hardware and software. The only way to make this economically feasible, however, was to use hardware that could a) interface with the accelerator the way the accelerator equipment was designed, and b) provide a straightforward interface to the new software (EPICS). And so we began exploring the use of programmable hardware solutions.

THE NEW LANSCE INDUSTRIAL I/O SYSTEM

For our first test case we decided to replace the Industrial I/O (IIO) portion of one RICE module with a commercial programmable logic controller (PLC). We defined "Industrial I/O" to encompass the basic, non-time-critical, non-closed-loop, and non-safety-critical functions of the control system. The PLC system worked well, but we discovered that it was not fast enough for some of our IIO binary output channels. It also could not time ADC reads to avoid the noise induced on the system by the accelerating RF.

For our next iteration, we traded the PLC for a National Instruments Compact RIO system, which is about as environmentally rugged as a PLC, but can also be several orders of magnitude faster. In the Compact RIO system, I/O cards plug directly into an FPGA. The FPGA can be programmed using LabVIEW, which gets translated into VHDL and then into the FPGA bitmap.

The standard RICE-replacement system we constructed is an 8-slot Compact RIO system that features 64 binary input channels, 32 sinking binary output channels, 8 solid state relay binary output channels, 32 analog input channels, 16 analog output (DAC) channels, and 4 stepper motor channels. The analog inputs can be triggered in order to avoid RF noise. Variants of the standard system are possible, and may replace some functionality (e.g. stepper motors) with other functionality (e.g. counters). In most cases, one (IIO) chassis can service all the industrial I/O channels in one RICE module. Details of the specific implementation can be found in the companion paper [2].

ADAPTING THE COMPACT RIO TO RICE

The specific features of the RICE system that we wanted to emulate in our new IIO controller were:

- Timing the ADC reads to avoid RF-induced noise
- Multiple protocols for binary output commands.
- Knob-friendly stepper motors.
- Easy on-line reconfiguration

*Work supported by the US Department of Energy under contract DE-AC52-06NA25396

Binary Output Protocols

The LANSCE control system uses four different command protocols for binary output channels. These are:

- **Command Only:** The simplest protocol. Turn it on and it goes on. Turn it off and it goes off.
- **Command With Latchback:** The most complex protocol and the most common protocol used by RICE. Each command channel has an associated readback channel. The command value tracks to the value of the readback channel until a command is issued. The command value is held for a specified "hold time", allowing time for the readback to reflect the new command value before the command channel starts tracking it again. Latchback channels are useful when a device can be commanded from multiple sources.
- **Momentary Normally Open:** The command channel is normally low. Writing a 0 to the channel has no effect. Writing a 1 to the channel causes the device to change state. When a 1 is written, the command channel will be held high for the specified hold time, after which it reverts to low. A "momentary normally open" channel with a latchback is useful for implementing fault/reset logic.
- **Momentary Normally Closed:** The same protocol as "momentary normally open" except that the command output is inverted.

CAMAC and VME systems require a different card type for each of four binary output protocols. Even more card types are required if the hold times are implemented in hardware. By programming the protocol into the controller, we were able to implement all the binary command channels with only two card types.

Control Knobs and Stepper Motors

One of the unique features of our original control system is its heavy reliance on stepper motors as the primary analog output device and assignable control knobs as the primary analog output interface. This was a problem when we started to integrate EPICS into the control system because EPICS is not very good at implementing control knobs and particularly bad at controlling stepper motors with control knobs. Consequently, we spent a lot of time adapting our EPICS system to work well with control knobs [3].

A problem that can occur when knobbing a stepper motor is that the pulses can accumulate faster than the motor can turn, resulting in overshooting the intended target. We avoid this problem by limiting the output of the control knobs to only the number of pulses that can be accumulated in a fifth of a second. We also programmed the stepper motor controller to preempt the current pulse stream whenever a new pulse stream is received. In this way we guarantee that the device will stop moving when the knob stops turning.

Instead of using a stepper motor card, we chose to implement our stepper motor controller with a simple binary output card. This allowed us to program in the exact pulse width, speed and ramping characteristics that the accelerator equipment expected from RICE.

Stem Cells and Reconfiguration

In the RICE system, when you needed to change the protocol of a binary output channel, or give it a different readback, you simply moved a jumper or re-routed a wire. Typically this could be accomplished in a matter of minutes and did not disturb any of the other equipment controlled by that RICE module. To accomplish this same task with an FPGA, first the FGPA source code must be changed. Then (in the case of Compact RIO) the source code must be translated into VHDL. Then the bitmap needs to be reconstructed from the VHDL (a potentially lengthy process). Finally, the Compact RIO system must be taken off-line while the bitmap is re-flashed and the Compact RIO rebooted.

Reconfiguration occurs more frequently than one who is not accustomed to the workings of an accelerator laboratory might think. In order to minimize the amount of time it takes to reconfigure our FPGA systems, we adopted a "Stem Cell" approach. Under this approach, the FPGA code for a binary output channel (for example) resembles a biological stem cell. A binary output "stem cell" has the possibility of becoming any command type (command only, latch-back, momentary open, or momentary closed), using any binary input for its readback, and having any hold time between 0 and 65 seconds (in millisecond intervals). The "stem cell" does not take on a specific function until it receives instructions from a configuration process. The configuration process reads a configuration file and assigns the specified functions to the stem cells. Configuration runs at startup, but can be invoked again at anytime. Thus it is possible to completely reconfigure the action and behavior of a binary output channel "in vivo" without any interruption of service.

The chief drawback of the "stem cell" approach is that it requires more FPGA real estate per channel. When we first tried this approach on a Virtex 2 FPGA, we only had enough space to implement 11 binary output channels. After upgrading to a Virtex 5, however, we had more than enough room for 40 binary output channels and 64 binary input channels.

Binary outputs are not the only stem cells in our system. We can also dynamically configure the dynamic range and trigger of our analog input channels, the pulse rate and (to some extent) the ramp up rate of our stepper motor channels, and the integration time of our counter channels.

EMBEDDING THE CONTROL SYSTEM IN THE HARDWARE

Perhaps one of the most dramatic ways to tailor the hardware to your control system is to actually embed it

within the hardware. Many commercial products such as PLCs, PACs, Serial Controllers, and beam diagnostics employ an embedded processor running a real-time or "near" real-time operating system. This raises the possibility of actually embedding the control system (or at least the "front end" part of the control system) in the device's processor and letting it interact directly with the vendor's code. One of the strengths of belonging to a control system collaboration is that vendors have been willing to entertain and even market this ability. We have already seen this with products from Instrumentation Technologies [4], Moxa [5], National Instruments [6], Yokogawa [7], and ZTEC Instruments [8].

The Compact RIO uses a power-PC running the vxWorks real-time operating system. National Instruments and CosyLab collaborated with us to install a complete EPICS I/O Controller (IOC) on the Compact RIO. The EPICS software runs concurrently with the National Instruments software and communicates with it using a shared memory interface concept originally pioneered at the Spallation Neutron Source [9].

One immediate advantage of embedding the control system is that data displayed on the operator screens now comes directly from the Compact RIO. With the PLC, the data had to go from the PLC to an EPICS IOC and then to the operator screen. Another advantage is that the Compact RIO is now able to take advantage of all the EPICS utilities such as archiving, bumpless reboot, access security, alarm handling, performance diagnostics, and our local software for making control knobs work well with EPICS. A final advantage is that the Compact RIO system may also access other local devices that have Ethernet interfaces – such as, for example, a PLC.

One final topic worth mentioning is that FPGAs are now coming equipped with hard and soft-core processors, making it possible to embed the control system right on the FPGA. Some examples can be found in [10], [11], and [12].

CONCLUSION

Not only has the Compact RIO IIO controller worked well as a replacement for RICE, but with some minor tweaks we found it also works well as a replacement for CAMAC. To date we have replaced two RICE modules and two CAMAC crates with our standard Compact RIO IIO systems. The three things that have contributed to this success have been 1) the ability to program the controllers, which allowed us to interface with existing accelerator equipment, 2) embedding the control system in the controller, which simplified the software interface, and 3) the ability to reconfigure the interface on-line.

ACKNOWLEDGEMENTS

The authors would like to thank David Bonal, Thierry Debelle and Ryan King of National Instruments for their invaluable support and assistance with the initial implementation of the Compact RIO IIO Controller. We would also like to thank Rok Šabjan of Cosylab for his initial work on getting EPICS to run on the Compact RIO as well as for the initial implementation of the shared memory library.

REFERENCES

[1] D.R. Machen, R. Gore and D. Weber, "A Compact Data Acquisition And Control Terminal For Particle Accelerators", PAC'69, IEEE Trans. Nucl. Sci. NS-16, p.883 (1969) ; http://www.JACoW.org

[2] S.A. Baily and E.Björklund, "Tailoring the Hardware to Your Control System", 5th NI Big Physics Symposium, Austin, August 2011, publication pending; https://decibel.ni.com/content/groups/big-physics?view=documents

[3] E. Björklund, "Toward A General Theory of Control Knobs," ICALEPCS'01, San Jose, November 2001, THAP071, p.608 (2001); http://www.JACoW.org

[4] C.Scafuri, V. Forchì, G. Gaio and N Leclercq, "Embedding a TANGO Device Into a Digital BPM", PCaPAC'06, Newport News, October 2006, p. 23; http://www.jlab.org/conferences/PCaPAC/PCaPAC2006_proceedings.pdf

[5] G.Y. Jiang and L.R. Shen, "An Embedded EPICS Controller Based on Ethernet/Serial Box," ICALEPCS'07, Knoxville, October 2007, WPPA06, p.328 (2007); http://www.JACoW.org

[6] E. Björklund, A. Veeramani and T. Debelle, "Using EPICS Enabled Industrial Hardware for Upgrading Control Systems," ICALEPCS'09, Kobe, October 2009, WEP078, p.555(2009); http://www.JACoW.org

[7] A. Uchiyama et. al., "Development of Embedded EPICS on F3RP61-2L," PCaPAC'08, Ljubljana, October 2008, WEX03, p.145 (2008); http://www.JACoW.org

[8] B.L. Shaw and C.D. Ziomek, "Off-The-Shelf EPICS Instrumentation for Remote Waveform Monitoring & Analysis," IPAC'10, Kyoto, May 2010, MOPE082, p.1173 (2010); http://www.JACoW.org

[9] D. Thompson and W. Blokland, "A Shared Memory Interface Between LabVIEW and EPICS," ICALEPCS'03, Gyeongju, October 2003, TU514, p.275 (2003); http://www.JACoW.org

[10] J. Weber, M. Chin, C. Timossi, and E. Williams, "Hardware and Software Development and Integration in an FPGA Embedded Processor Based Control System Module for the ALS," PAC'07, Albuquerque, June 2007, MOPAS031, p.503 (2007); http://www.JACoW.org

[11] A. Götz, J. Butanowicz, L. Slezak, C. Scafuri and G. Gaio, "Ubiquitous TANGO," ICALEPCS'07, Knoxville, October 2007, WPPA28, p.374 (2007); http://www.JACoW.org

[12] J. Odagiri et. al., "Fully Embedded EPICS-Based Control of Low Level RF System for SuperKEKB," IPAC'10, Kyoto, May 2010, WEPEB003, p.2686 (2010); http://www.JACoW.org

SUITABILITY ASSESSMENT OF OPC UA AS THE BACKBONE OF GROUND-BASED OBSERVATORY CONTROL SYSTEMS

W. Pessemier*, G. Raskin, H. Van Winckel, Institute of Astronomy, K.U.Leuven, Belgium
G. Deconinck, P. Saey, ESAT-ELECTA, K.U.Leuven, Belgium

Abstract

A common requirement of modern observatory control systems is to allow interaction between various heterogeneous subsystems in a transparent way. However, the integration of off-the-shelf (OTS) industrial products - such as Programmable Logic Controllers (PLCs) and Supervisory Control And Data Acquisition (SCADA) software - has long been hampered by the lack of an adequate interfacing method. With the advent of the Unified Architecture (UA) version of OPC (Object Linking and Embedding for Process Control), the limitations of the original industry-accepted interface are now lifted, and also much more functionality has been defined. In this paper the most important features of OPC UA are matched against the requirements of ground-based observatory control systems in general and in particular of the 1.2m Mercator Telescope. We investigate the opportunities of the "information modelling" idea behind OPC UA, which could allow an extensive standardization in the field of astronomical instrumentation, similar to the efforts emerging in several industry domains. Because OPC UA is designed for both horizontal and vertical integration of heterogeneous subsystems, we explore its capabilities to serve as the backbone of a dependable and scalable observatory control system, treating industrial components like PLCs no differently than custom software components. Performance measurements and tests with a sample of OTS OPC UA products are presented.

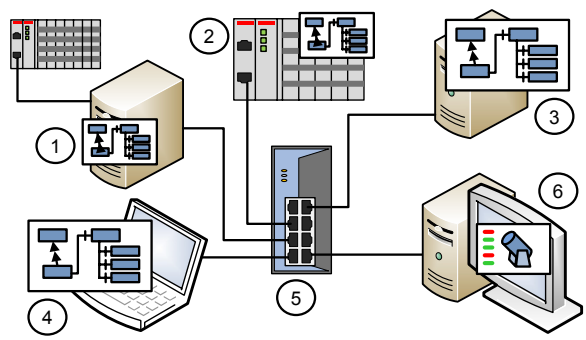
INTRODUCTION

Context and Motivation

Similar to the efforts in industry, control system engineers for astronomical observatories are continuously seeking to reduce development and maintenance costs and to increase system dependability. One of the most prominent trends is to foster software reusability, by creating frameworks which separate infrastructure code (i.e. the common *technical* aspects) from application logic (i.e. project-specific *functional* aspects) [1]. Another trend is the continued effort to integrate OTS (either commercial or public domain) software and hardware into these control systems [2]. Today, generic frameworks exist which meet both requirements to a large extent, even though some issues persist. Firstly, deeply involving frameworks may impose too strong constraints on the freedom of developers to choose the best fitting or best known technology for a given problem [3]. A "one size fits all" framework may exhibit code bloat and a lack of scalability, especially towards smaller and less demanding (embedded) applications. Secondly, as most generic frameworks to date rely on CORBA or DDS[1] as middleware to "glue" all subsystems together, integration of mainstream OTS industrial products remains inconvenient. As will be elaborated in this paper, OPC UA may offer a solution to these issues.

Methods

For this assessment we have compiled a list of qualitative and quantitative properties that are commonly required by observatory software frameworks, and in particular for the Belgian Mercator Telescope (a 1.2m optical telescope based at La Palma and currently being refurbished). The approach is twofold: not only do we want to verify if the OPC UA specification can satisfy these requirements, but we also want to test whether OTS implementations are complete and mature enough to build a functional infrastructure. For this purpose we have created a test set-up as depicted in Fig. 1, consisting of a small sample of commercial products with OPC UA connectivity. Naturally, many more products[2] of many more vendors are available on the market. Particularly important in the set-up is the SDK that we used to develop code for an experimental framework based on OPC UA. For this purpose we evaluated the C++ SDK by Unified Automation.



1. Kepware KEPServerEX UA server (www.kepware.com)
2. Beckhoff CX5020 PLC + UA server/client (www.beckhoff.com)
3. National Instruments LabVIEW UA server/client (www.ni.com)
4. Unified Automation C++ SDK server/client (www.unified-automation.com)
5. Moxa EDS-G308 Gigabit switch (www.moxa.com)
6. Siemens WinCC + Allmendinger UA client (www.siemens.com, www.allmendinger.de)

Figure 1: Test set-up.

* wim.pessemier@ster.kuleuven.be

[1] Object Management Group specifications: http://www.omg.org/spec.
[2] See OPC Foundation website: http://www.opcfoundation.org.

OPC UA BASICS

OPC UA is the successor of the suite of "classic" OPC standards that were developed by the OPC Foundation to interface industrial PLCs with SCADA and HMI (Human Machine Interface) systems. Besides the specification [4], a collection of code deliverables has been released to speed up application development and to facilitate compliance to the standard. Among these code deliverables are communication stacks in ANSI C/C++, JAVA, and C# .NET, which implement common lower level functionality. Two transport mechanisms are defined: the simple but high performance *UA TCP* and the Web Services standard *SOAP/HTTP*. Optional message signing and encryption is based on the *WS-SecureConversation* specification, and authentication is controlled by X.509 certificates. The stacks also implement data encoding via an efficient binary scheme or via XML. Built on top of the stack, an SDK is responsible for higher level functionality such as service handling and address space management. SDKs are offered by the OPC Foundation and by third party companies.

One of the most characteristic aspects of OPC UA is the extensive support for *information modelling*. Information entities (*Nodes*) and the relationships between them (*References*) can be defined, and exposed to the network in the *address space* of a server. Not only data itself but also meta-data can be exposed and transported over the wire. The semantics of the information model can be described formally in a *namespace*. A Node has several attributes and can represent an object, a variable, a method, an object/variable/data/reference type or a view (see [5]). A full mesh network of Nodes and References, representing a fictitious information model of METIS (the proposed Mid-Infrared E-ELT Imager and Spectrograph, see [6]) is shown in Fig. 2. The example is not meant to be optimal or complete, but attempts to illustrate some features of OPC UA. Object-oriented concepts such as data abstraction, encapsulation, polymorphism and inheritance are supported, as well as dynamic model changes, extensible references types, and cross-server referencing.

REQUIREMENTS

Table 2 lists a number of essential requirements [7], and whether they are met by the OPC UA specification and the tested SDK (second and third columns respectively). Only general comments are given, even though many requirements have more in-depth technical aspects which were verified by developing code snippets on top of the Unified Automation SDK. A recurring observation is the level of detail that is present in the OPC UA specifications and the derived stacks and SDKs. A substantial amount of these built-in features are distinctive for heterogeneous distributed control systems. For instance, data changes can be monitored asynchronously via the *MonitoredItems* services, which include standard arguments so that clients can request (and negotiate) the interval to sample the underlying data source, the publishing interval to minimize bandwidth overhead, an optional filter to deal with noisy data sources, and even a flag to request resampled data from the server. Dependability mechanisms are also strongly built into the OPC UA specification. For instance, OPC UA clients need to send "publish requests" to a server in order to keep the latter informed about the readiness of the client to receive events.

Quantitative requirements are more difficult to assess since they may vary strongly between projects and depend largely on the used hardware. We therefore use the set-up from Fig. 1 to measure the performance of a few typical service calls (*read* in this case) between a client and a server built with the Unified Automation SDK, see Table 1. The server is running on a Linux laptop and the client on a Windows PC. The UA Binary protocol is used to transmit the unencrypted and unsigned data via the gigabit network.

Table 1: Performance Measurements

Variables per call	Single UInt32 variable	1K*1K UInt32 matrix
1	0.37 ms	181 ms (∼22.1 MB/s)
10	0.68 ms	1784 ms (∼22.4 MB/s)
100	3.18 ms	
1000	23.26 ms	

The results indicate that the binary protocol of OPC UA is capable of high performance transmission of data (or meta-data) between components running on computer platforms. The performance in a more heterogeneous set-up (including embedded and real-time devices) is less useful to assess since it depends even more on the particular hardware and software configuration. Besides a priority mechanism for event handling on application level, no Quality of Service (QoS) features are specified by OPC UA. And, even though throughputs of around 22 MByte/s can be achieved when transmitting messages of several MByte (see Table 1), no special services are defined to exchange large data volumes. While other middleware technologies such as CORBA and DDS can meet strict real-time and data streaming requirements, OPC UA is clearly tailored for system architectures with complementary, special-purpose bus systems. These could include real-time fieldbuses, a bus for large data volume transport, an IEEE1588 timing bus, a safety bus, ... on a dedicated or shared medium (with managed switches to preserve QoS if needed).

FEASIBILITY

Besides accommodating special purpose frameworks (such as LabVIEW or PLC systems), an OPC UA-based infrastructure should provide a comfortable framework to develop applications in a popular language, on common platforms. Although SDKs provide useful high level functionality that is normally not part of a general middleware like CORBA, an extra software layer and tools are needed to simplify application development and deployment.

Table 2: Qualitative Analysis

Requirement	Spec.	SDK	Comments
Platform independence	Yes	Yes	No reliance on platform specific technology; SDKs are available for most platforms (non-real-time, real-time, and embedded).
Scalability	Yes	Yes	OPC UA is designed for vertical integration from sensor level up to mainframe level. Applications expose well defined *Profiles* to describe which services they support. In our test set-up, UA applications could be developed for embedded devices with few supported profiles (such as PLCs) to rich-featured servers on Linux PCs.
Reusability	Yes	Yes	OPC UA facilitates and encourages the definition and standardization of information models that extend the standard namespace. This is exemplified in Fig. 2: standardization can be achieved on a low level (e.g. by representing all sensor values according to the standard *AnalogItemType*), on a high level (e.g. by deriving a *DetectorControllerType*) or even on a system-wide level (by referencing Nodes in other servers in order to facilitate the organization of complex systems and avoid unnecessary aggregation).
Communication paradigms	Yes	Yes	Services are defined to read and write data, to invoke methods on remote objects, and to subscribe to events and data changes (monitored items). Multiple operations (e.g. method calls) can be invoked in one request. The tested SDK provides both a synchronous and asynchronous API for the service calls. Processes requiring a longer execution time should be modelled as *Programs* (standardized state machines) instead of methods.
Complex data	Yes	No	Servers need to expose the structured variable instances, their type definition and their encoding details (so that clients can discover how to interpret them). The SDK we tested required manual encoding of these user-defined types, but both stub/skeleton code generators (for types known at compile time) and generic helper classes (for types known at runtime) were due to be released with the next minor update.
Lifecycle management	No	No	Due to scalability and platform independence, OPC UA systems may be very heterogeneous. Therefore transparent starting, stopping and killing of client and server processes is complicated or (in case of embedded devices) may be impossible. Lifecycle management of objects similar to the container-component model can be accommodated well by OPC UA, but the central "manager" (or a hierarchy of managers) needs to be developed on top of the SDK. When a manager references Nodes in other servers, clients can easily resolve these Nodes by using standard OPC UA services.
Alarms	Yes	Yes	*Alarms* and *Conditions* (representing the state of a system) are standardized in OPC UA by extending the event mechanism. Advanced features such as "shelving" (to prevent operators from being flooded by alarms) are built-in.
Logging	Yes	Yes	Straight-forward to implement by extending the *BaseEventType* (containing fields for the timestamp, priority, description, ...) or one of its subtypes (e.g. for auditing purposes).
Location transparency	Yes	Yes	OPC UA foresees *Discovery Servers* at well known locations, to which other servers can register. Clients can query them to find the location of the server they need. More advanced discovery of applications is feasible via a *Global Directory Service*.
Historical archive	Yes	Yes	OPC UA does not define how historical information is stored, but does specify how it can be accessed. An attribute is assigned to variables and event notifiers to specify whether historical data or events are available. Clients can invoke standard services to read or modify raw (or even resampled) historical data at given timestamps and intervals.
Dependability	Yes	Yes	A wide range of mechanisms to increase reliability and availability are offered on a low level (e.g. sessions handlers can deal with network interruptions) and on a high level (e.g. compliance tools are available to test servers and clients). Security measures (confidentiality, integrity, authentication and authorization) are part of the stack and are therefore implemented by most OPC UA products. Due to the extensible design of OPC UA (e.g. new encodings and transport protocols can be added) and the wide support from industry, OPC UA is likely to be future proof. However, due to the lack of a standardized API, a system infrastructure may strongly depend on a particular stack or SDK.

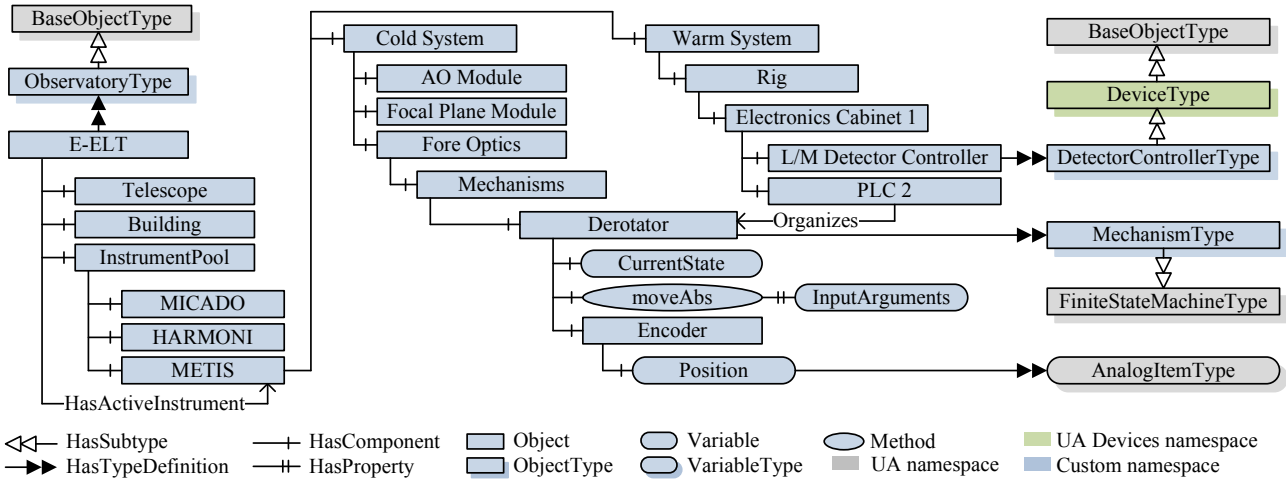


Figure 2: Example of a full mesh information model in the context of METIS.

An important difference with CORBA is the poor availability of stub and skeleton code generators. Similar to CORBA Interface Definition Language (IDL) compilers, OPC UA code generators can be used to convert XML namespace definitions into stub and skeleton code. Unfortunately, only a few OTS code generators existed at the time of writing this article. Those we did find were either inaccessible or too inflexible for us. Therefore we decided to estimate the expense of a homemade solution. For this purpose we wrote a few Python scripts which were able to convert XML namespace definitions (designed in UML [1] by one of the available graphical tools) into C++ server skeletons with rudimentary introspection and dispatching functionality. Application developers can derive classes from these skeletons and include business logic. When the corresponding objects are instantiated and registered to our server implementation, both the object instance and type definition are automatically exposed. On the client side we opted for a design with a narrow interface, in order to accommodate multiple actions in one service call. While it's clear that our experimental code is unsuitable to be used outside our test environment, we expect that it will take less than a man-year of work to prepare and commission a basic framework for the Mercator Telescope.

CONCLUSIONS AND OUTLOOK

In conclusion, we consider OPC UA suitable to serve as the "backbone" technology of ground-based observatory control systems. We estimate that the specification and available implementations are sufficiently mature and complete to cover the requirements of many projects. Even so, at this moment an infrastructure built around OPC UA cannot take full advantage from all opportunities offered by the specification, since the adoption by industrial products is still limited. For instance, UA Binary connectivity is generally supported, but SOAP/HTTP is not.

With respect to a "general purpose" middleware such as CORBA, OPC UA is naturally more tailored to facilitate interaction between the components of a heterogeneous and distributed control system. In effect, much of the basic functionality (alarms, logging, monitored items, ...) required by infrastructures for astronomical observatories is already standardized by the OPC UA specification. As a result, development efforts are reduced (because this functionality is offered by SDKs), and more importantly, industrial software and hardware can be integrated much more seamlessly since they comply to the same standard.

More effort and a more rigorous approach are certainly needed to create a framework which streamlines and facilitates application development for PCs, but we consider the investment worthwhile even within the constraints of a project on the scale of the 1.2m Mercator Telescope. For the refurbishment of this control system we aim to host most software on industrial PLCs and Linux PCs. We will thereby try to benefit as much as possible from technology advancements such as powerful and dependable soft-PLCs, flexible (object-oriented) PLC programming environments, and OPC UA as the universal control interface.

REFERENCES

[1] G. Chiozzi et al., "Enabling technologies and constraints for software sharing in large astronomy projects", Proc. SPIE, Vol. 7019, 2008.

[2] G. Chiozzi et al., "Trends in software for large astronomy projects", Proc. ICALEPCS, 2007.

[3] P. J. Young et al., "Instrument control software requirement specifications for Extremely Large Telescopes", Proc. SPIE, Vol. 7740, 2010.

[4] OPC Foundation, OPC UA specification Parts 1-13.

[5] W. Mahnke, Stefan-Helmut Leitner and Matthias Damm, "OPC Unified Architecture", Springer-Verlag, 2009.

[6] B. R. Brandl et al., METIS: The Mid-Infrared E-ELT IMager and Spectrograph, Proc. SPIE, Vol. 7014, 2008.

[7] G. Raffi, B. Glendenning, "ALMA Common Software Technical Requirements", ESO, Issue 1.0, 2000-06-06.

SNS ONLINE DISPLAY TECHNOLOGIES FOR EPICS*

K.U. Kasemir, X. Chen, J. Purcell, E. Danilova, ORNL, Oak Ridge, TN 37831, U.S.A.

Abstract

The ubiquitousness of web clients from personal computers to cell phones results in a growing demand for web-based access to control system data. At the Oak Ridge National Laboratory Spallation Neutron Source (SNS) we have investigated different technical approaches to provide read access to data in the Experimental Physics and Industrial Control System (EPICS) for a wide variety of web client devices. We compare them in terms of requirements, performance and ease of maintenance.

INTRODUCTION

Traditional control system operator interface tools were often standalone applications for a specific operating system [1]. Recent developments emphasized operating system portability, allowing users in their offices or even at home to run the same applications that are used in the control room [2]. These new tools are highly integrated, offering a workflow that previously required switching between multiple standalone programs [3]. While there is a continuing need for such feature-rich applications, there is at the same time a desire for web-based control system access. This online view of the control system might be read-only, at a limited update rate and restricted to a subset of the machine, but it should be accessible from any web browser, be it a desktop computer, laptop, netbook or cell phone, without need to install any additional software on the client device.

CONTROL SYSTEM DATA

The following examples refer to EPICS, but the fundamental ideas apply to any control system. The control system allows network access to data points called Process Variables (PVs). Certain PVs might change at a known rate, but the most common case is data that changes at arbitrary times. "Set point" PVs for example will stay constant until adjusted by an operator. PVs for temperature readings might vary slowly over time, while other machine parameters can change rapidly.

Polling vs. Event-Driven

A control system display tool should be event-driven, reflecting changes as they occur. If the display is polling the PVs for updates, resources are wasted polling data points that stay constant while at the same time missing updates that are faster than the poll rate.

BASIC WEB TECHNOLOGY

The Hypertext Transfer Protocol, HTTP [4], is the basis of all web technology. It typically functions as follows:

1) Web client establishes a network connection to the web server.
2) Web client requests a web page by sending GET /some_web_page.html HTTP/1.1
3) Web server returns the web page content.
4) Server and client close the network connection.

The request as well as the response can include additional parameters. Server and client may keep the network connection open for follow-up requests, but the protocol remains a request-response pattern.

There is no method in HTTP that is implemented by all web clients to support a "push" of updates from the server to the client. Every data transfer needs to start with a request sent by the client. This makes HTTP fundamentally unsuitable for event-driven updates as desired for control system displays! Users would have to manually initiate a request by pushing the "Reload" button on their web browser to update the display to the most recent data.

There are several commonly used approaches to work around this limitation in a way that is supported by the majority of web browsers.

Web Browser Plug-Ins

Most web browsers allow the installation of plug-ins to extend their functionality. The CAML project uses a WebCA plug-in to add native support for EPICS Channel Access to several web browsers [5]. This solution offers the best possible performance because the web client directly receives updates from the control system.

The approach is, however, specific to certain web browsers. The direct network traffic between the client and the control system is also often blocked by firewalls that only allow access to the standard web server port TCP 80, or generally restrict access to the control system to computers on the plant network.

Consequently, this approach cannot offer generic web access to control system displays from a wide range of devices, including cell phones.

Ajax

Ajax [6] uses JavaScript code running inside the web browser to create XMLHttpRequest (XHR) objects [7] that in turn perform requests to a web server. The JavaScript code then handles the response, typically by updating the web page with the received data. Ajax still uses the HTTP request-response pattern, but it no longer requires the user to *manually* initiate requests because the JavaScript code inside the web browser itself can initiate the requests.

Ajax can be used to implement a polling web client that periodically requests updates from the web server.

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy

Long Poll

A "Long Poll" [8] uses a polling Ajax web client as just described with the additional agreement between server and client that the server can delay its response until updates are available:

1) JavaScript in client starts XHR to request.
2) Web server waits until updates become available.
3) Once there are updates, web server returns the data.
4) Web client displays the data and immediately starts another XHR, i.e. repeat step 1.

Technically, this is still a polling request-response mechanism. In a true event mechanism the client would not have to re-issue any request. The server would simply send updates via a once established network connection.

Performance is lower than a true event mechanism because of the additional network traffic for sending the requests. The client or server might even close their network connection at the end of each transaction, requiring a new connection for each long poll.

To the end user, there is hardly a difference: The display can be updated as soon as the server sends new data.

SNS STATUS WEB SITE

The SNS Status web site is a Java Server Page (JSP) project running on a Tomcat web application server [9].

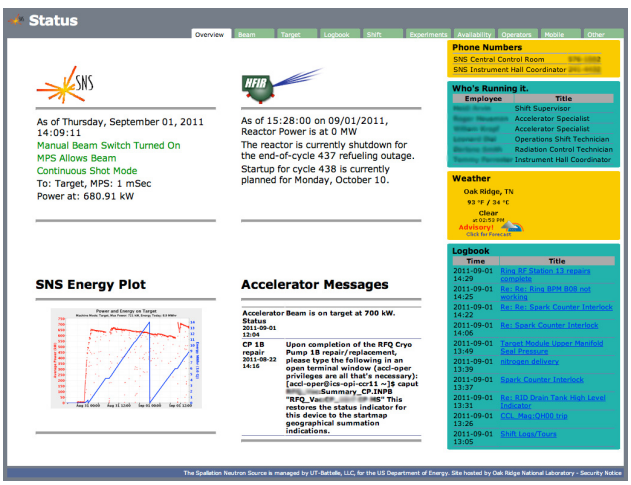


Figure 1: SNS Status Web Page.

It displays a fixed set of pages, arranged in "tabs", to summarize the recent history of SNS operation, including beam power, state of beam lines, availability statistics, recent logbook messages, and shift summaries.

Certain sections of these web pages are periodically updated via Ajax. The beam power in the upper left section of Fig. 1 for example is refreshed every 3 seconds, while other sections of the page are updated at intervals of 30 seconds or even minutes.

Java code on the web server, i.e. Tomcat, subscribes to a predefined, fixed list of PVs and maintains a table of their current values. Ajax requests from web clients invoke Tomcat servlets, which then read the current PV data and format it to for example generate the beam status information shown in Fig. 1.

The SNS Status Web site has been online since July 2010. It has been very stable. It can be accessed by any web client, and requires only a few CPU and memory resources. Sections of this web site are displayed on hallway monitors throughout the SNS facility. The beam history information from this web site is a key component of daily status meetings.

The majority of its content, however, is based on data read from a relational database, not online PVs. Only about 50 PVs are monitored by the Tomcat application, updating sections of the web page at comparably slow periodic rates as mentioned.

The web site offers a common denominator of data that is of interest to many users. The overall content and its layout are fixed. End users cannot customize it to meet their specific, individual needs.

SNS DASHBOARD

The SNS Dashboard was created to provide more flexibility. Users can choose from a list of widgets, some of which are customizable. Widgets can be opened and closed; they can be "dragged" to the desired position on the web page.

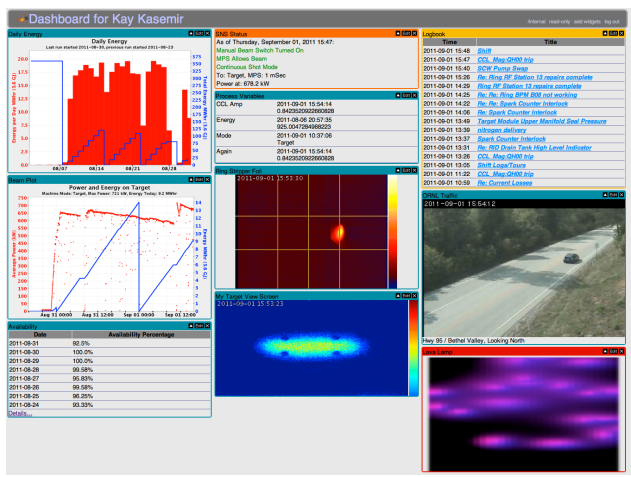


Figure 2: SNS Dashboard.

At this time there are about 20 widgets. Some duplicate information that is already offered on the SNS Status web site, but users can now consolidate the information of interest on their personal dashboard. There are also widgets for a user base that was too small to warrant inclusion in the SNS Status web site. Examples include a widget that lists the devices affected by scheduled power outages, or a widget with indicators for the status of certain groups of front-end computers.

Widgets that rely on PV data utilize a Long Poll, updating as events are received from the control system. The PV pool is not fixed as in the SNS Status web site. Some widgets allow the user to configure which PVs to display. The web client will invoke a subscription servlet

for these PVs. The web server subscribes to the control system PVs and returns updates as they arrive from the control system in the next Long Poll.

In our initial implementation, each widget that depends on PV updates would issue a Long Poll, but we learned that most web browsers limit the number of parallel XHR requests to 6 or even 2. To adjust for this limitation, our dashboard software performs only one Long Poll for the whole dashboard page of a user. The data returned by the web server contains tags, which are used by the client to dispatch the data to the widgets that requested it.

The achievable Long Poll update rate varies. 10 Hz are often possible, but we throttle it to 1 Hz because it was considered fast enough for a generic online display. The web server tracks the PV subscriptions of each web client. If a PV changed, it is marked for update. Adding a 1 second delay to the Long Poll means that the web server can often return the accumulated updates for multiple PVs within one Long Poll, reducing the network traffic.

The Dashboard software is more complex than the SNS Status web page. The dashboard configuration for each user is persisted in a relational database. It is read by the web server when a user logs on, then duplicated in the web client. This duplication in the web client allows the user to efficiently edit the configuration inside the web browser. When the user relocates a widget inside the web browser or adds a PV name to a widget, the web client does not need to communicate each mouse click or key press to the web server. The web client only sends the end result to the web server, for example the new widget location or the added PV name, thereby minimizing the network traffic and optimizing the responsiveness of the web page.

Keeping the dashboard data consistent between the web server and client requires attention to detail. To complicate the implementation, this has to be done in two very different programming environments: Java on the server, but JavaScript, a Document Object Model (DOM) and Style Sheets in the web client. The JQuery scripting library [10] simplified the client-side implementation, especially the Ajax handling for different web browsers.

The Dashboard was published in June 2011 and is gaining acceptance. Users can create multiple setups, for example one for the desktop and one for the cell phone. We have been adding new widgets based on user requests. At this time, however, users can not create their own widgets or contribute them to the list of available widgets.

CSS WEB OPI

Most of our recent developments for control room and office computer user interfaces were based on the Eclipse Rich Client Platform (RCP) [2, 3]. RCP is a powerful Java framework that allows the implementation of applications, which then run on different operating systems like Microsoft Windows, Apple Mac OS X and Linux. The RCP Standard Widget Toolkit (SWT) library implements the actual user interface components like windows, buttons and other graphical elements that the end user sees on her computer screen.

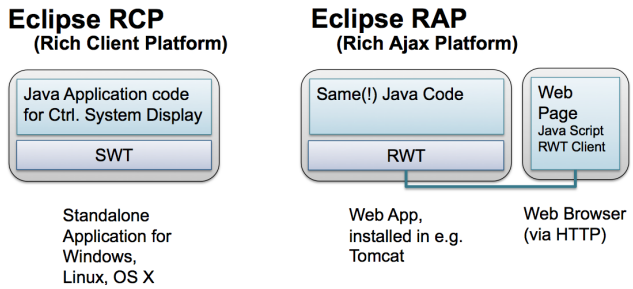


Figure 3: Comparison of RCP and RAP.

The Eclipse Rich Ajax Platform (RAP) project offers an RWT library that replaces SWT [11]. Ideally, the same Java application code that was originally developed for SWT will also run with RWT, but instead of creating a display on the local computer screen, as was the case for RCP, the RAP application is acting as a web server. Web clients that connect to this web server will see a display in their web browser that mimics the original SWT display.

The RAP application is typically installed in a web container like Tomcat. The RWT library creates HTML and Java Script code for the web client, using a Long Poll mechanism to send updates to the web client. An important point, however, is that the application developer can almost completely ignore the details of Tomcat, HTML, JavaScript, Ajax, Long Poll, because this is handled by RAP.

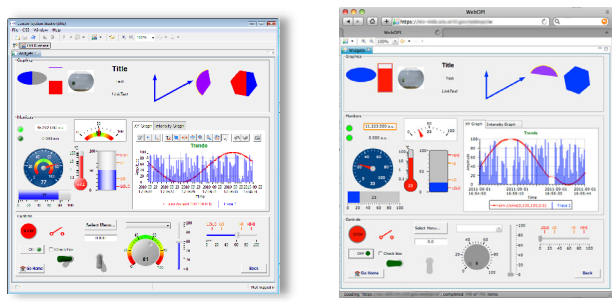


Figure 4: CSS 'BOY' display (left) and corresponding Web OPI representation.

The CSS operator interface BOY [12] was modified to execute not only with RCP but also RAP. As shown in Fig. 4, existing display files that were created with BOY can now also be executed in a web browser, providing an almost identical look and behavior. Users can develop any type of BOY display and then not only use that in the control room or their office computer but also at home and on portable device, including smart phones.

To port the existing RCP version of BOY to RAP, code for reading display files, connecting to EPICS network channels and handling control system events remained the same. This code is now executed within the web server. Its performance is naturally identical to the same code running in the RCP version.

Calls into SWT, which originally updated a physical display, now go into RWT, which in turn generates JavaScript that is queued for delivery to the web client via Long Polls. The JavaScript generated by RWT causes the web client to update the display inside the web browser.

The direct display update as performed by RCP is naturally more efficient than the RAP implementation necessitated by the shortcomings of HTTP. On a computer where an RCP display for 100 text widgets updating at 10 Hz uses about 12% of the CPU and 50MB of RAM, the combination of RAP under Tomcat with Firefox as a web browser used about 40% CPU and 80MB of memory.

When increasing the number of text widgets to 1000, the RCP display would continue to update at 10Hz, while the RAP version would degrade to 3Hz.

In an operational setup, the CPU load is of course spread across different computers. The web server can be a high-performing server machine, while the clients can be an office PC as well as a much simpler smart phone. The server handles most of the logic except for the actual display. As long as there are not too many events from the control system that require display updates, small devices like smart phones can display operator screens of considerable complexity.

The RAP design aims to allow "single sourcing". The same source code should seamlessly compile for both RCP and RAP. The main difference between the two is the fact that RCP code targets a single display, while RAP needs to handle multiple (virtual) displays. Since the SWT programming interface associates fonts, colors and other user interface resources with a display, the BOY code needed to be extended to track such resources for multiple (virtual) displays.

The network traffic between web server and client adds a certain delay to RAP that is not present in an RCP application, including the extreme case that the web client can be closed without directly notifying the web server. RAP can only detect this via time-outs.

While the BOY RCP code and the Web OPI RAP code are currently not fully single-sourced, the majority of the source code is shared by both approaches. Most important, the Web OPI offers all the features of the RCP BOY display without having to perform any coding on the HTML or JavaScript level.

We presented the web OPI to SNS users in August 2011. Its current implementation relies on certain pre-release snapshots of RAP code because RAP is still maturing technology, but our initial experience has been very good.

SUMMARY

The core web technology, HTTP, is less than ideal for online control system displays. Appropriate use of Ajax, especially the Long Poll paradigm, can alleviate fundamental HTTP limitations.

The SNS Status web uses basic Ajax technology to generate generic displays for a wide audience. The Dashboard uses Long Poll and more client-side JavaScript to offer more customization and faster updates for users that need specialized displays. The Web OPI uses RAP for web access to any BOY display, offering utmost flexibility because users can create their own BOY displays in CSS.

These three approaches complement each other. Users can access generic status displays with zero effort, invest time in creating their fully customized displays for the Web OPI, or use the Dashboard as an intermediate solution.

The Web OPI explores the limits of HTTP-based control system displays. A true event mechanism that can transfer binary data could encode the display updates more efficiently. In the future, WebSockets [13] might be available in all web clients, allowing for more efficient control system displays.

REFERENCES

[1] "MEDM", "StripTool", "ALH" and other network client tools for EPICS, http://aps.anl.gov/epics/extensions
[2] K. Kasemir, "New User Interface Capabilities for Control Systems", PAC 2009, Vancouver, Canada
[3] K. Kasemir, "The Best Ever Alarm System Toolkit", ICALEPCS 2009, Kobe, Japan
[4] HTTP - Hypertext_Transfer_Protocol Overview, http://www.w3.org/Protocols/
[5] Th. Pelaia, "CAML and Web CA Status", EPICS Collaboration Meeting, Legnaro, Italy, 2008
[6] Ajax (programming), August 2011, http://en.wikipedia.org/wiki/Ajax_(programming)
[7] XMLHttpRequest Specification Candidate Recommendation 3, August 2010, http://www.w3.org/TR/XMLHttpRequest
[8] Comet (programming), August 2011, http://en.wikipedia.org/wiki/Comet_(programming)
[9] Tomcat web application server, http://tomcat.apache.org
[10] JQuery Scripting Library, August 2010, http://jquery.com
[11] Eclipse Rich Ajax Platform, August 2010, http://www.eclipse.org/rap
[12] X. Chen, K. Kasemir, "BOY, a Modern Graphical Operator Interface Editor and Runtime". PAC 2011, New York, NY
[13] The WebSocket API, Draft 25 August 2011, http://dev.w3.org/html5/websockets

THE WONDERLAND OF OPERATING THE ALICE EXPERIMENT

A. Augustinus, P. Chochula, L. Jirdén, M. Lechman, P. Rosinský, CERN, Geneva, Switzerland,
O. Pinazza, INFN Sezione di Bologna, Bologna, Italy and CERN,
G. De Cataldo, INFN Sezione di Bari, Bari, Italy and CERN,
A. Kurepin, INR - RAS Moscow, Russia and CERN,
A. Moreno, Universidad Politécnica de Madrid, ETSI Industriales, Madrid, Spain.

Abstract

ALICE is one of the experiments at the Large Hadron Collider (LHC), CERN, Geneva, Switzerland. Composed of 18 sub-detectors each with numerous subsystems that need to be controlled and operated in a safe and efficient way. The Detector Control System (DCS) is the key to this and has been used by detector experts with success during the commissioning of the individual detectors. During the transition from commissioning to operation, more and more tasks were transferred from detector experts to central operators. By the end of the 2010 data-taking campaign, the ALICE experiment was run by a small crew of central operators, with only a single controls operator. The transition from expert to non-expert operation constituted a real challenge in terms of tools, documentation and training. A relatively high turnover and diversity in the operator crew that is specific to the HEP experiment environment (as opposed to the more stable operation crews for accelerators) made this challenge even bigger. This paper describes the original architectural choices that were made and the key components that enabled the DCS to come to an homogeneous control system that would allow for efficient centralized operation. Challenges and specific constraints that apply to the operation of a large complex experiment are described. Emphasis will be put on the tools and procedures that were implemented to allow the transition from local detector expert operation during commissioning and early operation, to efficient centralized operation by a small operator crew not necessarily consisting of experts.

INTRODUCTION

ALICE (A Large Ion Collider Experiment) is a general purpose heavy–ion detector installed on the 27 km Large Hadron Collider (LHC) at CERN. The experiment is designed to study the physics of strongly interacting matter and the quark–gluon plasma in nucleus–nucleus collisions. Data-taking during proton-proton runs provides reference data for the heavy-ion programme and addresses a number of specific strong-interaction topics for which ALICE is complementary to the other LHC experiments.

The experiment (Fig. 1) is composed of 18 sub-detectors and the collaboration currently involves over 1300 physicists, engineers and technicians from 116 institutes in 33 countries. The overall dimension of the detector is 16x16x26m3 with a total weight of approximately 10 000 tons.

The operation of the experiment relies on several independent online systems, each responsible for a specific domain of the operation:

- The Detector Control System (DCS) for the control and safety of the experiment; this paper will concentrate on this system
- The Data Acquisition (DAQ) system is responsible for the readout of the physics data, for event building and for data transport.
- The Trigger (TRG) system selects the interesting events and triggers the readout of the experiment.
- The High-Level Trigger (HLT) system performs online reconstruction of data in order to reject or tag events and to allow for data compression.

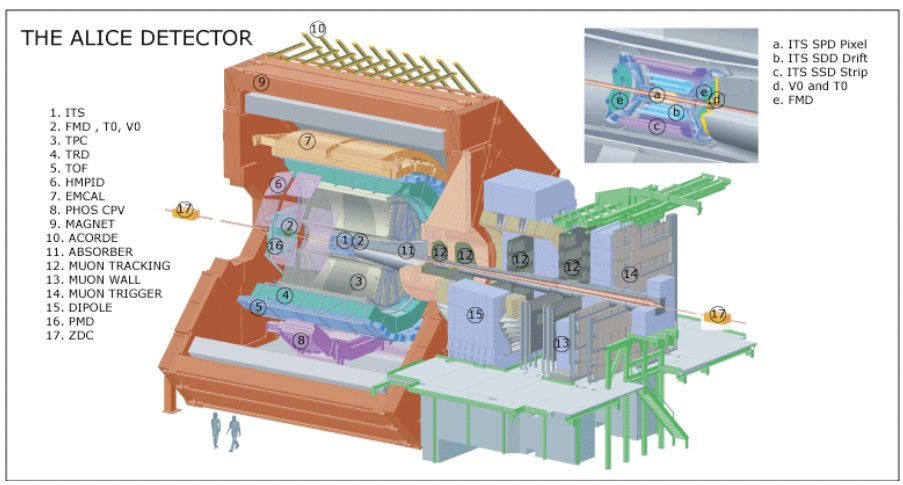


Figure 1: The ALICE detector at the LHC.

THE DCS IN ALICE

The main task of the Detector Control System in ALICE is to ensure safe and efficient operation of the experiment [1]. It provides configuration, remote control, and monitoring of all experimental equipment to allow the entire experiment to be operated from the ALICE Control Room (ACR) at LHC point 2, through a unique set of operator panels. The DCS provides the optimal operational conditions so that the physics data taken by the experiment is of the highest quality. The control system has been designed to reduce the downtime of the experiment to a minimum and hence contribute to a high running efficiency. It also aims to maximise the number of readout channels operational at any time.

The sub-detector control systems are provided by the contributing institutes; this work of over 100 developers from all around the world and from various backgrounds is coordinated by a small central controls team. The core of the controls system is a commercial Supervisory Control and Data Acquisition (SCADA) system: PVSSII [2]. It controls and monitors the detector devices, provides configuration data from the configuration database, and archives acquired values in the archival database. It allows for data exchange with external services and systems through a standardized set of interfaces.

In order to complement the PVSSII functionalities, a software framework has been built around PVSSII. It provides tools and guidelines for simplified implementation of the detector control systems. The core of this framework is built as a common effort between the LHC experiments, in the context of the Joint COntrols Project (JCOP) [3]. The main tools cover Finite State Machine (FSM), alarm handling, configuration, archiving, access control, user interfaces, data exchange, and communication. To cater for specific ALICE needs, the JCOP framework is complemented by components specific to ALICE. The complete ALICE framework is used by the sub-detector experts to build their own control applications, such as high voltage control, front-end electronics control, etc. Well over 100 such applications are finally integrated for a large and global ALICE control system.

FROM INSTALLATION TO ROUTINE OPERATION

After a long design, R&D and construction phase, installation of the first sub-detectors at the experiment site started in 2006. With this also the installation of the DCS infrastructure started, allowing the sub-detector groups to install their control systems.

From this moment, the time up to 2008 was heavily used to test and commission all sub-detectors to prepare for the first collisions in the autumn of 2008.

The year before resuming operation, late 2009 was used to install and commission more sub-detector modules, with the associated equipment and controls. This time was also used to gain experience in all aspects of running the experiment. The experiment collected cosmic ray data during several months, gearing up for the restart in October 2009.

After the end of year stop, operation resumed early 2010 to finish with a dedicated heavy-ion run. The last end of year stop saw again major installation work, notably the installation of a complete sub-detector (EM calorimeter).

Early this year, 2011, operation resumed, and in parallel, the commissioning of the newly installed detector equipment commenced.

Up to and including the start of 2009, the operation of the experiment was in the hands of the numerous sub-detector experts. It was only after several months of operation in 2010 that operation started to become more centralized. The transition to a fully centralized operation, where the whole experiment was run with a handful of people in a shift crew, took nearly 1 year. From early summer this year, the experiment is routinely run with a shift crew of 5.

It is clear that the evolution of the experiment has an impact on the DCS. The DCS has to follow the evolution to be able to fulfil its task of ensuring safe and efficient operation. The following sections will describe some of the challenges.

THE EVOLUTION CHALLENGE

Evolution in the Use and Users of the DCS

The early years of operation concentrated on detector debugging and commissioning. The DCS was used, and still developed, locally by the various sub-detector groups, providing all the functionality needed by the experts to operate their equipment for their test programs. The main task of the DCS at that moment was remote control of equipment, with only little emphasis on protection (e.g. from operator errors) since the system was exclusively used by system experts. The user interfaces used in this phase give access to the level of single channel operation of the equipment, and give the user a very detailed view of the system.

With the start of data-taking, the emphasis shifted more on operational aspects than pure remote control. Rather than manipulating individual channels, the simultaneous operation of groups of channels or whole sub-systems became important. Although several sub-detectors were operated at the same time, the actual operation was performed locally by the various sub-detector operators. These operators usually had a good knowledge of their sub-detector, but were not necessarily experts. Due to this, the need arose to present these operators with dedicated interfaces that would allow grouped operation and present information in a more concise form. At this point the protection against operation errors also became more important.

With the transition to more centralized operation, the tasks previously executed locally by the sub-detector operator were now carried out by a central operator.

Because of the large number of sub-detectors, it was obvious that the central operator would need additional tools to efficiently operate all sub-detectors. In addition, these operators usually have very little experience in sub-detector operation, so the tools shall also hide any complexity in sub-detector operation.

DCS Follows the Evolution of the Experiment

Another challenge for the DCS is that it has to cope with a continuously evolving experiment.

- Regularly, additional detector modules are installed that need to be integrated into the existing controls.
- Existing equipment is exchanged or upgraded. The DCS will have to be able to integrate this with minimal effort.
- While gaining experience in operation, the DCS might have to adapt to the newly gained insights.

The architecture of the DCS allows for meeting these challenges. The system was built with scalability in mind; new systems can easily be added. The notion of partitioning (taking control of a sub-tree in the controls hierarchy) was a key feature of the system to allow concurrent independent operation of sub-detectors.

THE SINGLE OPERATOR CHALLENGE

With the transition to a central operation of the experiment DCS, there were several challenges. Up to this point each detector was operated by a sub-detector operator; people with good knowledge of their sub-detector. This mode of operation required a shift crew of 25, which is not sustainable for the collaboration in the long term. More and more operation responsibilities were transferred to the central shift crew and currently a crew of 5 is running the experiment.

Unlike the accelerator sector, where shifts are usually covered by a reduced team of professionals, large HEP experiments face a specific challenge. It is common practice to encourage a maximum number of people to experience the operation of an experiment. In this way, people that will analyse physics data gain a better understanding of how operational issues can influence data quality. Nowadays large experiments are collaborations of a large number of institutes, and covering experiment shift duties are expected to be fairly shared between the collaborating institutes. This is the reason that the experiment is run with a relatively large pool of people that will only cover a small number of shifts. To illustrate, for 2011, the 926 8 hour ALICE DCS operator shifts are covered by about 80 different operators. So, an average operator will not do more than 11-12 shifts, making it difficult to gain solid experience.

This fact has a clear impact on the experiment DCS.

Training

The large number of new operators that need to be trained by the DCS experts puts a non-negligible load on the limited resources of the central team. For the future, a partly web based operator training might be envisaged, given the positive experience with such training courses at CERN.

Documentation and Instructions

Clear, extensive, and explicit documentation is fundamental to guide the majority of operators that have only limited experience. Classical (word processor produced) documentation is used for static documentation, and for more volatile instructions, Twiki is used.

As the central operator is responsible for the operation of all sub-detectors, the recovery of anomalies and errors is the primary task of the central operator. To facilitate this, sub-detectors have to provide a set of instructions for events that can be recovered by the central operator. One can easily understand that collecting these instructions from such a large number of sources is a managerial challenge.

This large collection of instructions shall be made easily accessible for the operator. To achieve this, context sensitive access to instructions is integrated into the main tools and interfaces used by the operator.

Hiding Complexity

Obviously, the central DCS operator cannot have the detailed knowledge of all the operational details of all sub-detectors, therefore central operation is only possible when the complexity and specificity of each sub-detector is hidden for the operator.

Each sub-detector is represented by a limited set of states (e.g. ready for data-taking, standby, etc.) and a limited set of generic commands (e.g. go ready, switch off, etc.). The calculation of the state and the internal processing of the generic commands are programmed in Finite State Machine logic by the sub-detector experts.

Experience has shown that it is nowadays relatively uncommon to operate a single sub-detector; they are typically operated in 3-4 groups. Therefore, recently, a new tool (Fig. 2) has been introduced to allow the operator to operate groups of detectors; the configuration of the groups can be done by experts.

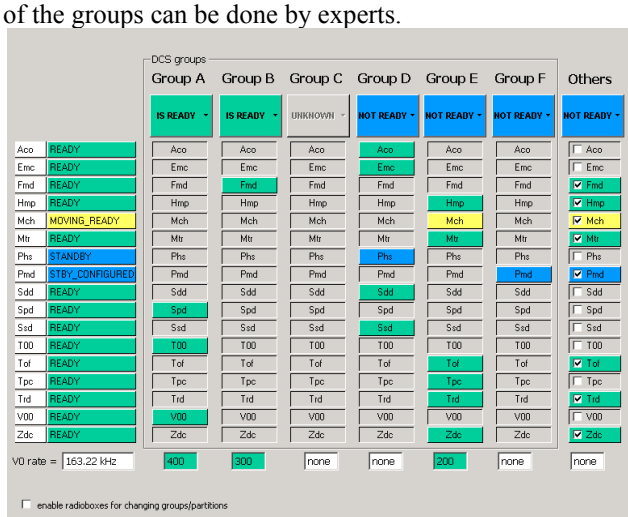


Figure 2: Group operation.

Automation

The majority of tasks the operators must perform currently require active intervention. This implies the operator has to judge when and what action to perform (e.g. switching off certain sub-detectors when beam conditions change). In order to aim for a more coherent operation we try to automate as much as possible in any routine operation.

Currently a tool is being prototyped that allows an expert to define automatic actions triggered by external events (such as a beam mode change). The tool will be accompanied with an interface where the operator can follow the automatic actions that are performed.

THE COORDINATION CHALLENGE

Integration

Each sub-detector DCS is developed under the responsibility of that sub-detector, by their experts. In order to guarantee integration of all these separate developments into a single, homogeneous control system, a strong coordination of the developments and integration is needed. At a very early stage - with the start of the first developments - strong guidelines were issued and regular reviews performed to monitor the progress.

Also now, as development still continues, the strict coordination continues to ensure the correct integration of all sub-detector systems.

Maintenance and Development

All control applications are developed and maintained by the sub-detector teams. As with many of these projects in the HEP community, application development is often done by programmers with limited duration contracts, or that spend only a limited time in the project.

With the original author no longer available, long term maintenance or new developments are more and more delicate. In some projects, the applications are now maintained by the 3rd or 4th generation developer.

FUTURE

The sub-detector details are hidden from the central operator, and all standard operations are done with dedicated tools or interfaces. However, for error recovery procedures, the operator might be instructed to access more detailed sub-detector interfaces. In order to ease the navigation through such expert interfaces, and to limit the risk of operator errors, an effort shall be undertaken to encourage further uniformity of these interfaces (e.g. same look and feel).

Error recovery instructions are created by the relevant sub-detectors using templates, so that at least the form is homogeneous. However, there is currently only a limited check on the content of these instructions. A more systematic quality assurance of these instructions is currently under study.

CONCLUSION

The Detector Control System of the ALICE experiment was designed with flexibility in mind. It was with the start of a more centralized operation that the full power of the architecture of the DCS was unveiled and this architecture allows it to cope with the challenges it faces. Some of these challenges are typical for HEP experiments (as opposed to accelerator control), such as a relatively dynamic environment, and many inexperienced operators due to the high turnaround. The challenges are not only technical, but also managerial; a strong coordination closely following the activities in the sub-detectors is the key to preserving the homogeneity of the DCS.

REFERENCES

[1] ALICE Collaboration, Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger and Control System, CERN/LHCC/2003-062.

[2] ETM website http://www.etm.at/index_e.asp (the product was recently rebranded as “Simatic WinCC Open Architecture”)

[3] Holme, O. et al., “The JCOP framework,” proceedings ICALEPCS 2005, Geneva, Switzerland, 2005

PURPOSE AND BENEFIT OF CONTROL SYSTEM TRAINING FOR OPERATORS

Elke Zimoch, Andreas Luedeke, PSI, Villigen, Switzerland.

Abstract

The complexity of accelerators is ever increasing and today it is typical that a large number of feedback loops are implemented, based on sophisticated models which describe the underlying physics. Despite this increased complexity the machine operators must still effectively monitor and supervise the desired behavior of the accelerator. This is not alone sufficient; additionally, the correct operation of the control system must also be verified. This is not always easy since the structure, design, and performance of the control system is usually not visualized and is often hidden to the operator. To better deal with this situation operators need some knowledge of the control system in order to react properly in the case of problems. In this paper we will present the approach of the Paul Scherrer Institute (PSI) for operator control system training and discuss its benefits.

INTRODUCTION

For the scope of this paper a machine operator is someone who monitors and supervises the behavior of one or more accelerators from a control room. Typically, the only connection between the operator and the physical machine (which can be hundreds of meters away) is the control system with the computer network it is implemented on. In the case of PSI the operators are responsible for four different accelerators each with its own control systems, most of them based on EPICS.

The general task of the operators is to remotely control the accelerators. During normal machine operation this narrows down to setting up and optimizing the accelerator settings according to the scientific needs on one hand and handling system misbehavior and faults on the other hand.

As there is no other connection between the operator and the accelerator all this has to be done through the control system, or more precisely through the user interfaces of the control system.

MENTAL MODELS

In a perfect world the control system is transparent to the operator. The operator would only need to know about the accelerator. All relevant information would be provided by the control system, correctly and in the right way. Unfortunately, there is no perfect world. Even the "right way" to display data is different from one user to the next and from one use case to the next.

If the control system can not represent the accelerator correctly, it is not transparent to the operator. This implies that information is changed on its way from the accelerator through the control system to the operator. The data is distorted by the control system like a picture is distorted by a colored lens. Therefore, the operator needs some knowledge about the control system and how it works in order to be able to recognize the distortion and clean up the information.

In cognitive science and psychology the internal human representation of a complex external system is called a "mental model" [1]. We develop such models to help us handle the complexity of the real world. They assist us in simulating alternative behavior before applying it. And in addition, they allow reasoning about the way the systems will behave or develop [2].

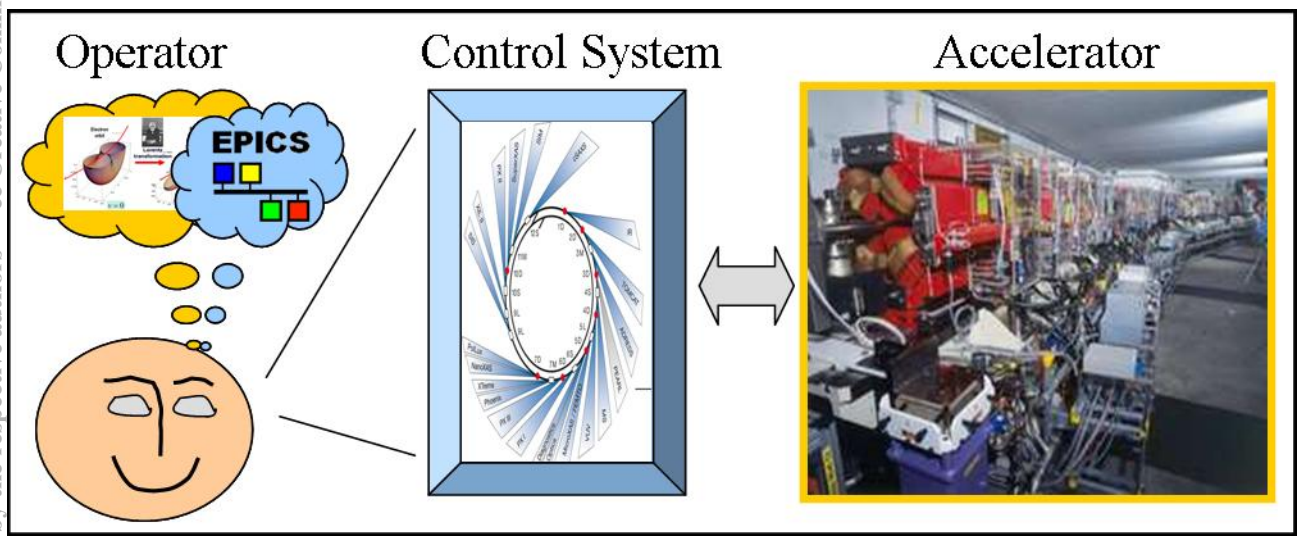


Figure 1: Operator relationship to Accelerator and Control System.

Unfortunately, to make a mental model work and fit into the "working memory" of a human brain, it has to be simplified. The aspects that are represented are influenced by background knowledge and developed over time with experience.

In the context of accelerator control the operator has to maintain two mental models: one for the accelerator and another for the control system of that particular accelerator. As the main task of the operator involves operating accelerators the mental model of the accelerator is usually acknowledged by everyone. For the majority of operator tasks this might be sufficient but as soon as some unusual situations appear the mental model of the control system gets more important. Especially during error diagnosis and recovery it is vital for the operator to distinguish between errors in the accelerator and errors in the control system.

CONTROL SYSTEM TRAINING

The question is how to influence the mental model operators have about the control system in such a way that it represents the actual reality. Fortunately, mental models are not static but change over time through learning and experience [3]. This led us to a twofold approach for operator training for the Swiss Light Source (SLS) at PSI.

EPICS Training Course

The first step prepares the ground and provides the theoretical background information. This is done in the "EPICS Training at PSI" course [4]. It consists of a lecture part and afterwards hands-on training. The lecture concentrates on an overview of the control system and informs about technical terms and general ideas.

We found that it is important to provide this lecture in German, the native language of the operators at PSI. Presenting unfamiliar concepts in a foreign language makes it more difficult to learn and accept these concepts. For scientists who do not speak German fluently the same course is available in English as it is not feasible to provide it in the native language for everyone.

Directly after the lecture a hands-on training is provided. Each operator has a dedicated control system console available as well as some hardware connected to a server. They are tutored to configure the server so that they can remotely control the hardware from the console. Some exercises are provided in order to channel the training but there are no constraints to what the operators can do.

It has been proven crucial for the success of the training to provide a safe playground and make the operators aware of it. Otherwise the fear of doing something wrong or making a mistake hobbles the playful approach. Without pressure the operators can find out how the control system reacts to faults and discover what to do in fault situations in their own speed of learning.

As the speed of learning can vary a lot given the diverse backgrounds of course participants, it is important for the trainer to have small groups. The PSI EPICS Training is provided once per month but only for four people at a time. This ensures that the trainer can concentrate on each student. In addition, each student having his own computer and hardware does not allow hiding behind someone who knows more or is quicker to understand.

Training through Experience

As mental models change over time they can decay. This happens if the knowledge of the operator is no longer up to date due to changes in the system, or if knowledge simply gets forgotten. In addition the build up of a mental model is quite slow for new operators if there is no possibility to gain experience.

In this aspect the excellent availability of the SLS has proven to be a problem. Faults occur too seldom to keep the mental models up to date. Because of this, mental models of operators (both, the accelerator model as well as the control system model) are subject to decay. But as the accelerator model gets some stimulation from change of setups and optimization of the facility it is less affected.

To counteract this knowledge decay we have introduced the "SLS Operator Training": once per month during scheduled machine time the trainer creates a problem with the actual accelerator facility. The problem is created by a dedicated program called "sabotage" that randomly selects one of several possible error sources and applies it to the accelerator or control system. One operator is chosen to solve the problem while the other operators and the trainer watch and provide help if needed.

In addition to providing experience in fault handling, the operator training allows to reflect on the way problems appear through the filter of the control system. Even if the induced fault is not connected to the control system the operator gets practical knowledge on how the control system shows problems and where to look for indications of misbehaviour.

The operators that observe these problem solving sessions have the possibility to compare the approach of their colleague to their own approach.

WHAT ELSE CAN BE DONE

To support mental models about the control system the functionality and status of the system itself should be visualized. Such user interfaces allow the operator to supervise the control system more explicitly than only through the behaviour of accelerator parameters.

Examples could be the status of network connections or hardware servers (called IOCs in EPICS). The EPICS framework provides already some tools for this purpose: the connection status of each parameter is available to the client automatically (but still needs to be displayed) and the vxStats package evaluates the status of an IOC. But error messages and status information of control system internals are often obscure. To avoid confusion, explanations or a glossary has to be displayed as well.

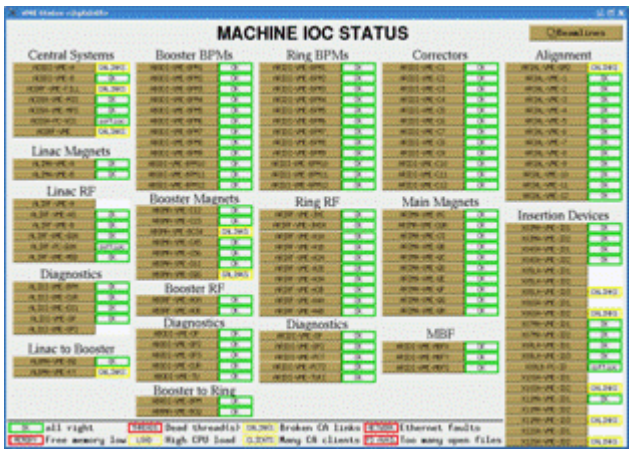


Figure 2: Example of a status screen for visualisation of Control System functionality.

Including control system information in error messaging systems like the EPICS "alarmhandler" can help the operators to distinguish between accelerator errors and control system errors. But it can hide the difference as well. Therefore, the implementation has to be thought through carefully.

CONCLUSION

Operators need mental models of the control system to recognize fault states and react appropriate to errors and misbehavior of both, the accelerator and the control system itself. Mental models gained only on infrequent experience can be imprecise or plain wrong in worst case. Control system training can provide a foundation to build better mental models and therefore help to enhance operator responses and machine availability.
For a refinement of the mental model repeated experience is needed. This can be provided by trainings sessions at the real accelerator. Reflection and discussion of the monitored error handling approach can optimize the impact.

ACKNOWLEDGEMENTS

We would like to thank

- Detlef Vermeulen and Anton Mezger for their support of the different trainings.
- The whole Controls Section for help and feedback.
- The PSI Operators for their patience.

REFERENCES

[1] P. N. Johnson-Laird, "Mental models", Cambridge University Press, 1983, Cambridge, UK.
[2] K. J. W. Craik, "The nature of explanation", Cambridge University Press, 1943, Cambridge, UK.
[3] N. A. Jones, H. Ross, T. Lynam, P. Perez, and A. Leitch, "Mental models: an interdisciplinary synthesis of theory and methods", Ecology and Society 16(1): 46, (2011), [online] URL: http://www.ecologyandsociety.org/vol16/iss1/art46/
[4] http://epics.web.psi.ch/training

JDDD, A STATE-OF-THE-ART SOLUTION FOR CONTROL PANEL DEVELOPMENT

E. Sombrowski, A. Petrosyan, K. Rehlich, W. Schütte, DESY Hamburg, Germany

Abstract

Software for graphical user interfaces to control systems may be developed as a rich or thin client. The thin client approach has the advantage that anyone can create and modify control system panels without specific skills in software programming.

The Java DOOCS Data Display, jddd [1,2,3,4], is based on the thin client interaction model. It provides inclusion of panel components and channel address inheritance for the creation of generic displays. Wildcard operations and regular expression filters are used to customize the graphics content at runtime, e.g. in a dynamic list component the parameters have to be painted only once in edit mode and then are automatically displayed multiple times for all available instances in run mode. This paper will describe the benefits of using jddd for control panel design as an alternative to rich client development.

INTRODUCTION

DESY is currently building a new 3.4 km-long X-ray free electron laser XFEL. This facility will deliver ultra-short light pulses with a peak power up to 100 GW and a wavelength down to 0.1 nm. The commissioning is planned in 2014. The software development has already started and concepts are currently being evaluated and improved at the FLASH FEL facility.

Jddd is developed as the main tool for the creation of graphical displays for the XFEL accelerator. It uses the thin client concept. In the following paragraphs advantages and disadvantages of thin and rich clients will be discussed.

THIN CLIENT VERSUS RICH CLIENT APPROACH

When implementing a client/server architecture it has to be determined if it will be the client or the server that handles the bulk of workload.

In a rich client system, a significant amount of data processing is done on the client or desktop system, while relatively little is done on the server. In contrast, a thin client generally does as little processing as possible on the client side and relies on accessing the server each time input data needs to be processed or validated.

Each of these approaches has its own benefits:

Advantages of Thin Clients

- The interface between client and server is clearly defined.
- Control system panels are rapidly developed using a set of predefined components/widgets.
- No programming skills are needed. Engineers, technicians and operators are able to design their own control panels. This reduces the commissioning time because the hardware and software development is in one hand.
- Panels are easily adapted to changing server properties and new hardware. The design has high flexibility to later modifications.
- Thin client software provides a standard look&feel and uniform functionality for all control panels independent from the developer.

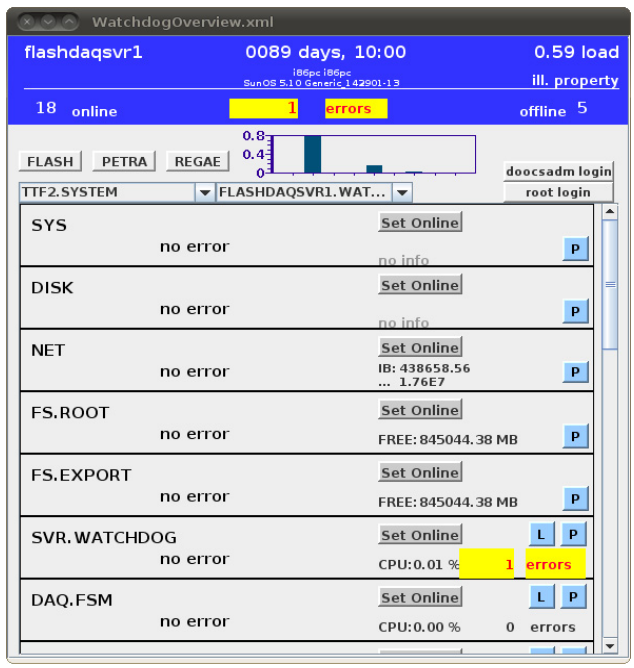


Figure 1: Screenshot of the watchdog overview panel in run mode. A dynamic list component in the lower part of the panel displays the status of all available servers on a selected CPU. A location chooser component, which is placed above the list, allows to change the facility and CPU address and therefore the list content. The buttons labelled FLASH, PETRA, REGAE switch between the different facilities.

Advantages of Rich Clients

- The functionality is not restricted to generic possibilities of thin client software. The display design is more user-friendly. Graphical components can be adapted to special use cases.
- Sometimes it's difficult to put all functionality in a server. Using rich clients, simple mathematical operations or even complex data analysis might be included on the client side.

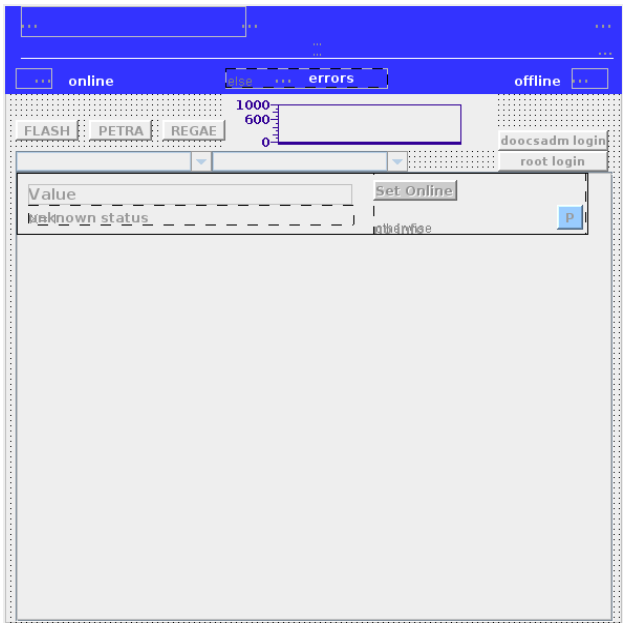


Figure 2: Screenshot of the watchdog overview panel in edit mode. The lower part of the panel contains a dynamic list component. This list is filled in run mode as displayed in Figure 1.

ADVANCED COMPONENTS AND FEATURES

To compensate the disadvantages of thin clients, to reach more flexibility in control panel design and to implement data manipulation, jddd uses different approaches:

- Dynamic components, which change their graphical content according to server properties.
- Logic components (if and switch component) switch the view between different layered panes (cases) depending on a user defined control system value.
- Buttons with “Set Component Property” function change e.g. include files or the panels base address at runtime.
- Wildcard operations, regular expression filters and JavaScript are used for mathematical data operations.

DYNAMIC COMPONENTS

Dynamic Lists

The dynamic list is a layered pane which can be filled with any jddd components in edit mode. It is assigned a base address with a wildcard for the facility, device, location, or property. At runtime the wildcard is replaced by all available addresses and the list is automatically filled with the required number of components. Figure 1 and 2 show an example for using the dynamic list in a watchdog overview panel.

Dynamic Includes

The dynamic includes component is a transparent layered pane which displays automatically multiple include components at dedicated positions according to their Z_POS and X_POS. In the FLASH orbit feedback example (Figure 4) the dynamic includes component shoes all available BPMs and steerer magnets.

LOGIC COMPONENTS

If and Switch Component

The if component contains two cases which consist of two different layered panes. The switch component contains an arbitrary number of cases. Depending on a user defined control system value one of these layered panes is displayed at run mode.

The condition for each case of a logic component is displayed in the jddd component inspector. Any case may contain nested logic components (see Figure 3).

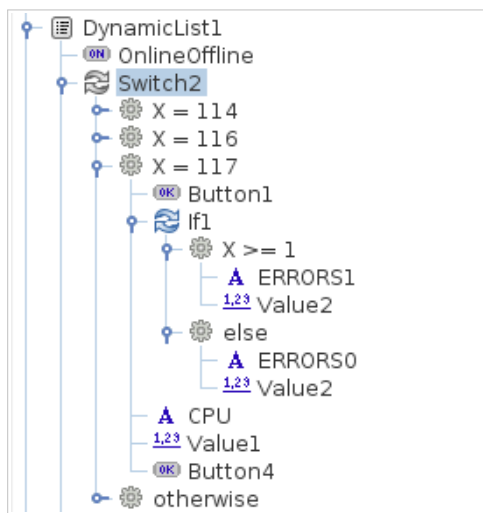


Figure 3: Example for nested switch and if components displayed in the jddd component inspector.

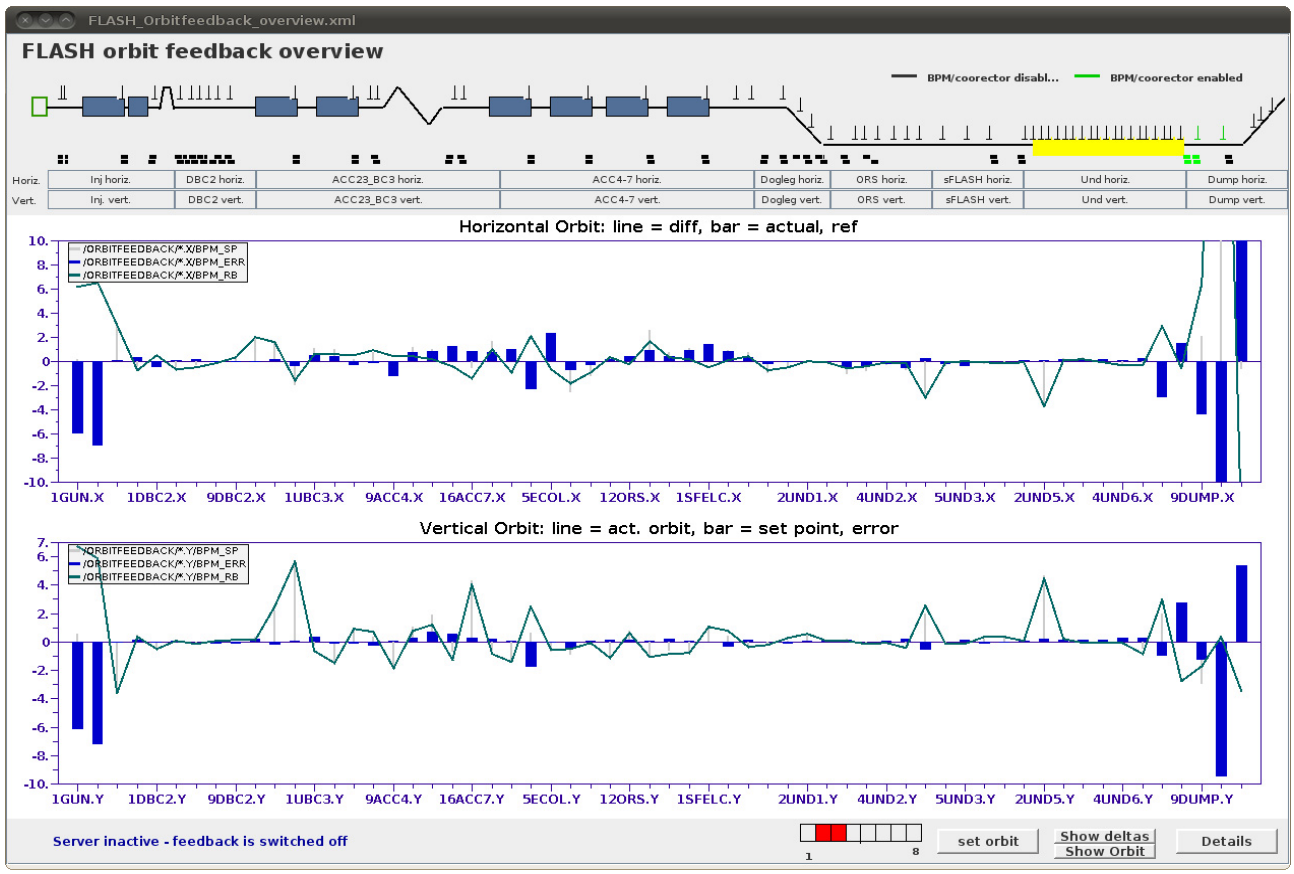


Figure 4: Screenshot of the FLASH orbit feedback panel. In the upper part of the panel dynamic include components are used for displaying BPMs and steerer magnets. The buttons “Show Deltas” and “Show Orbit” use the “Set Component Property” function to switch between orbit and delta plots.

SPECIAL BUTTON FUNCTIONALITY

A jddd button can have different functions. One of them is the “Set Component Property” function. Pressing this type of button one or multiple properties of any component in the panel are changed.

One possible use case is to modify the XML file of an include component. This changes the graphical content of the panel at runtime. In the FLASH orbit feedback (Figure 4) this button function is used to switch between delta and orbit view.

Another use case is to set the panels base address at runtime. In Figure 1 the FLASH, PETRA and REGAE buttons change the base address of the Watchdog panel and therefore switch the view between the different facilities.

WILDCARDS AND FILTERS

To allow the creation of dynamic control panels, which adapt the displayed information according to the selected server properties, jddd uses wildcard operations and regular expression filters.

Wildcards

To specify the content of dynamic lists, dynamic includes and location plots, wildcard operations in the address string are used. For example: If the address of a location plot is

TTF2.DIAG/ORBITFEEDBACK/*.X/BPM_XYZ_RB

all available locations ending with “.X” will be displayed in the plot (see upper location plot in Figure 4). For further filtering regular expression filters may be used.

Filters

For dynamic lists, dynamic includes, location chooser and the properties table a regular expression filter is available to filter the displayed data. It uses standard Java syntax for regular expressions, like *(?<!_SVR) to remove all locations ending with the letters “_SVR”. This filtering mechanism allows to display a dedicated subset of data.

JAVA SCRIPT

The "dataOp" property is used to perform mathematical operations with control system values using JavaScript syntax. It is available for the jddd components button, value, progress bar, coloured indicator, status register and switch. An example how to use this property is e.g.:

dataOp = $address1*2+$adddress2

At runtime "$address1" and "$address2" are replaced by the current values read from the specified addresses.

CONCLUSION

Jddd is more than an editor for simple thin clients. With logic components, “Set Component Property” buttons, filters and JavaScript support it offers advanced possibilities for nearly rich client development. Most panels with complex display logic can be implemented with this software and only a few high-level displays have to be created as rich clients.

Dynamic components and the use of wildcards ensure that displays do not have to be reworked if new devices are added to the control system.

Using jddd with its easy-to-use editor the development time for graphical displays is minimized. Engineers, technicians and operators are able to design control panels themselves, which reduces the workload of the control group. All panels will have a common look and feel and functionality independent from the developer.

REFERENCES

[1] http://jddd.desy.de
[2] E. Sombrowski, P. Gessler, J. Meyer, A. Petrosyan, K. Rehlich, “jddd in action”, ICALEPCS'09, Kobe, Japan, October 2009.
[3] E. Sombrowski, K. Rehlich, “First Experiences with jddd for Petra Vacuum Controls”, PCAPAC’08, Ljubljana, Slovenia, October 2008.
[4] E. Sombrowski, A. Petrosyan, K. Rehlich, P.Tege, "jddd: A Java Doocs Data Display for the XFEL", ICALEPCS'07, Knoxville, Tennessee, October 2007.

FIRST OPERATION OF THE WIDE-AREA REMOTE EXPERIMENT SYSTEM

Y. Furukawa, K. Hasegawa and G. Ueno*)
SPring-8/JASRI, Kouto, Sayo-cho Hyogo, 679-5198, Japan.
*SPring-8/Riken, Kouto, Sayo-cho Hyogo, 679-5148 Japan.

Abstract

The Wide-area Remote Experiment System (WRES) at SPring-8 has been successfully developed [1]. The system communicates with remote users on the basis of SSL/TLS with bi-directional authentication to avoid interference from unauthorized access to the system. The system has a message-filtering system to allow remote users access only to the corresponding beamline equipment and safety interlock system. This is to protect persons inside the experimental station from injury from any accidental motion of heavy equipment. The system also has a video streaming system to monitor samples or experimental equipment. We have tested the system from the point of view of safety, stability, reliability etc. and successfully performed the first experiment from a remote site, i.e., RIKEN's Wako campus, which is 480 km away from SPring-8, at the end of October 2010.

INTRODUCTION

Remote experiments brings many benefits to the users of a publicly opened experiment facility. One of the benefits is that the system frees users from travelling to and from the facility and users can spend that time on their creative work. Additionally, the system enables international collaboration during its 24-h operation: collaboration teams can operate their experimental equipment for 24 h, despite each collaborator working only during his or her day-time.

To realize a remote experiment system, we have to take into radiation and other safety and security issues. As previously reported [1], we have built a safe and secure Wide-Are Remote Experiment System (WRES) for remote experiment infrastructure. We have built a remote-experiment graphical user interface (GUI) and experiment control server for protein-crystallography experiments on the WRES, and after many tests, we have successfully completed the first operation from a remote site, i.e., RIKEN's WAKO campus 480 km far from SPring-8, at the end of October 2010. In this paper, we describe the remote experiment control system and summarize the first and following remote experiments.

WIDE-AREA REMOTE EXPERIMENT SYSTEM (WRES)

The wide-area remote experiment system (WRES) has been developed for safe and secure remote experiment at SPring-8 [1]. For safe and secure remote experiments, we have to take into account 1) radiation safety, 2) other human safety, i.e., human physical safety, and 3) security for access over the Internet.

During a remote experiment, the situation for radiation safety is no different from ordinary experiments at SPring-8; therefore, we do not require special treatment for a remote experiment on radiation safety issues.

If a remote experiment user, who has limited information on the experimental station, operates experiment equipment while SPring-8 staff is accessing the equipment, heavy or high-speed equipment might hurt the staffs. This leads to the issue of human physical safety. To prevent this, we use the fact that when an X-ray beam is introduced into the experimental hutch that covers the equipment to shield users from X-rays in the hutch, the radiation-safety interlock system ensures there are no people in the hutch. We built a local interlock for human physical safety. The local interlock system only permits remote experiments when the hutch is "normally closed," i.e., the hutch is ready for introducing X-rays.

For network security we use Secure Sockets Layer / Transport Layer Security (SSL/TLS) communication with bi-directional authentication (Fig.1). SPring-8 staff issues an authentication certificate including a beamtime ID for the remote experiment and sends it to the remote user by e-mail. The authentication certificate is an electric file locked by a randomly generated password. The password is also sent to the remote user by postal mail. The user who has both the authentication certificate and password can start the communication with the remote experiment connection server at the SPring-8.

The connection server verifies the certificate and passes the messages from the remote user during the beam-time. From the beam-time ID included in the certificate the connection server obtains the allocated beam-time.

The messages exchanged between the remote user and the SPring-8 station control system are formatted as SVOC which is used in the SPring-8 control system "MADOCA." The connection server filters the messages and passes only permitted messages to the station control system.

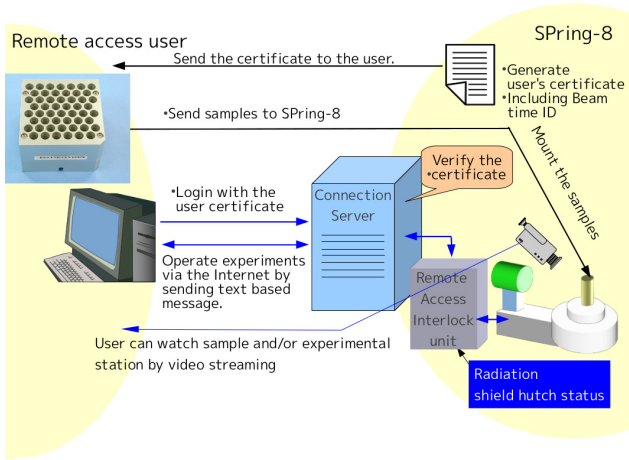


Figure 1: A schematic of the SPring-8 remote experiment system.

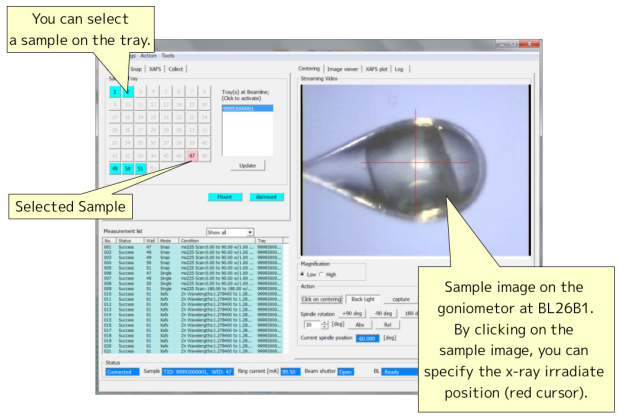


Figure 2: A graphical user interface for the remote user.

PROTEIN CRYSTALLOGRAPHY BEAMLINE BL26B1

In this section we briefly describe the SPring-8 protein crystallography beamline, BL26B1 [2]. The BL26B1 is one of the standardized protein crystallography beamline in SPring-8. The X-ray source is a bending magnet and the beamline optics consist of a double crystal x-ray monochromator, slits, X-ray mirrors. A monochromatized X-ray beam is introduced into the experimental hutch, which encloses the experimental equipment, consisting of X-ray slits, shutters, a goniometer, an X-ray CCD detector, and a sample changer. The sample changer is called SPACE [3].

The beamline and experimental station are highly automated, containing a sample changer. After users set their sample tray, which can contain up to 52 crystals on the sample changer, they can operate all their experiments from the software, called BSS [4], including sample screening, X-ray absorption spectra measurement, and automatically obtaining diffraction data.

Collected data are stored at a common data storage location for the protein crystallography beamlines and users can access their data via the Internet on the Web-based data distribution system called DCha [5].

Crystal growth becomes difficult as protein crystallography advances; therefore crystals become smaller and heterogeneous. It becomes important that the researcher who has grown the crystal specify the X-ray irradiated position for obtaining data because selection of the X-ray irradiated position determines data quality.

We focused on how a remote user can specify the X-ray irradiated position for the sample in a remote experiment. As shown in Fig.2, users select a sample number and SPACE mounts the selected sample. The sample image is then displayed on the remote user's GUI using a video streaming system, as shown at the top left on Fig.2.

By clicking on the video image, the remote user can specify the X-ray irradiated position. After taking several diffraction images, the user decides whether the sample quality is good enough for obtaining data. After the user sets the parameters for obtaining data, including the judgement of sample quality and the X-ray irradiated position, BSS starts automatic data collection. After all the data has been collected, the remote user can obtain the collected data using DCha.

FIRST OPERATION OF THE REMOTE EXPERIMENT

The first operation was made from RIKEN Wako campus, which is located 480 km from SPring-8 as shown in Fig.3. SPring-8 and RIKEN's Wako campus were connected to SINET-3 [6], which was the Japanese academic Internet backbone with 40-Gbps bandwidth[1]. RIKEN Wako campus was connected to SINET-3 at 10-Gbps bandwidht, and SPring-8 was connected to SINET-3 at 1-Gbps bandwidth.

The first experiment was successful and all the functions required for the remote experiment mostly worked well. The video streaming went down several times but this was later improved by upgrading ffserver and ffmpeg, which were used for the video streaming.

Latency from clicking on the video image to sample motion was less than 1 s. This was not dependent on the geometric distance because most of the latency was result of video encoding.

Video streaming was VGA size, 10 fps and streaming data rate was up to 800 kbps. Message exchanging data rate was much smaller than the video streaming data rate.

We confirmed that the quality of the collected data was the same as that of the data collected during an ordinary experiment.

[1]SINET was upgraded SINET-4 in March 2011 and SPring-8 connection bandwidth to the SINET will be increased up to 10 Gbps later.

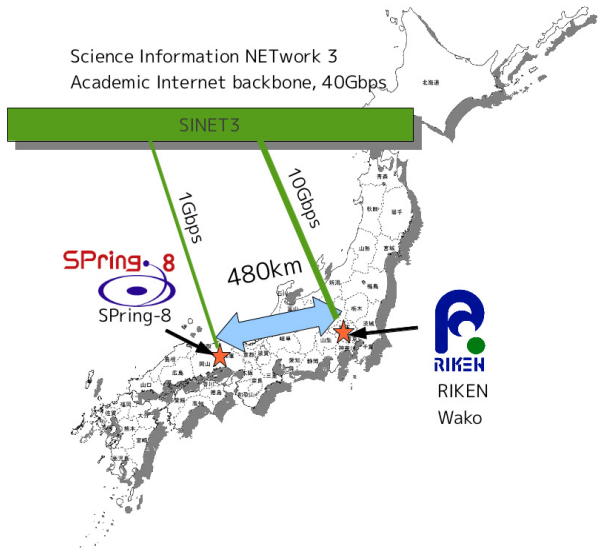


Figure 3: Location of RIKEN Wako campus and SPring-8.

CURRENT STATUS

The remote users interface was improved according to suggestions from the first experiment and opened for public protein crystallography users. We conducted training for the remote protein-crystallography experiment at the end of July 2011. We can now perform remote experiment on six beamlines.

We are now preparing for the first international remote experiment from Taiwan on BL12B2, which was constructed by the National Synchrotron Radiation Research Center (NSRRC) in Taiwan for protein crystallography.

We are also preparing a remote experiment for the another X-ray experiment field, X-ray Absorption Fine Spectra (XAFS) and plan to start testing coming April.

CONCLUSION

The first remote experiment based on the Wide-area Remote Experiment System was successful, and this indicates that the system is not only safe and secure enough but also efficient enough for conducting remote experiments. The SPring-8 remote experiment system will be expanded to include international remote experiments and other X-ray experiment fields.

REFERENCES

[1] Y.Furukawa, K.Hasegawa, D.Maeda, G.Ueno, "Development of remote experiment system", Proc. ICALEPCS 2009(Kobe, Japan) P.615

[2] Ueno G, Kanda H, Hirose R, Ida K, Kumasaka T, Yamamoto M, "RIKEN structural genomics beamlines at the SPring-8; high throughput protein crystallography with automated beamline operation" J Struct Funct Genomics. 2006 Mar;7(1):15-22

[3] Ueno G, Hirose R, Ida K, Kumasaka T, Yamamoto M, "Sample management system for a vast amount of frozen crystals at SPring-8" J Appl. Cryst. 2004 Dec;37(Pt 6):867-873

[4] Ueno G, Kanda H, Kumasaka T, Yamamoto M, "Beamline Scheduling Software: administration software for automatic operation of the RIKEN structural genomics beamlines at SPring-8" J Synchrotron Radiat. 2005 May;12(Pt3):380-4

[5] Okazaki N, Hasegawa K, Ueno G, Murakami H, Kumasaka T, Yamamoto M, "Mail-in data collection at SPring-8 protein crystallography beamlines" J.Synchrotron Radiat. 2008; 15:288-291

[6] http://www.sinet.ad.jp

[7] http://www.srrc.gov.tw

COGNITIVE ERGONOMICS OF OPERATIONAL TOOLS

A. Luedeke, PSI, Villigen, Switzerland

Abstract

Control systems have become increasingly more powerful over the past decades. The availability of high data throughput and sophisticated graphical interactions has opened a variety of new possibilities. But has this helped to provide intuitive, easy to use applications to simplify the operation of modern large scale accelerator facilities? We will discuss what makes an application useful to operation and what is necessary to make a tool easy to use. We will show that even the implementation of a small number of simple application design rules can help to create ergonomic operational tools. The author is convinced that such tools do indeed help to achieve higher beam availability and better beam performance at all accelerator facilities.

INTRODUCTION

A "tool" is something that helps you to achieve a goal. It is called an "ergonomic tool" if it has been tailored to be used by humans, taking into account our physical and cognitive limitations. These days the operator in an accelerator control room pursues his goals exclusively by means of computer applications. The discipline that deals with the design of ergonomic computer applications is human-computer interaction (HCI). For the software industry this field has continuously gained in importance over the last few decades. The branch of human-computer interaction that deals specifically with the analysis of cognitive processes is known as cognitive ergonomics. It deals, for example with diagnosis, decision making and planning: all required of operators in modern industries. At the same time human-computer interaction has been widely ignored in the development of modern control systems for accelerators. The intention of this article is to advertise the basic methods of cognitive ergonomics to develop ergonomic operational tools for the control of accelerators.

MOTIVATION

One of the most important figures of merit for accelerator operation managers is beam availability. Figure 1 shows the contributions of the different systems to beam interruption time for the Swiss Light Source (SLS) in 2009. There appears to be no contribution from "bad operational tools", but "operator faults" do account for 3% of the downtime. The operational applications at the SLS are the only means of the operator to interact with the accelerator; therefore an operator fault is most likely an unintended action of the operator: he did not properly predict the consequence of his action. Or we could say that the application failed to communicate the risks of the action to the operator in a timely manner.

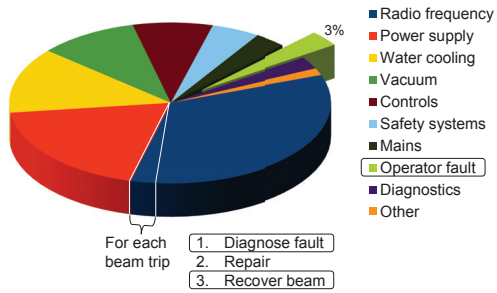


Figure 1: Beam failure statistics of the Swiss Light Source for 2009. Operational tools do affect the number of operator faults and the time to diagnose faults and recover beam.

The design of ergonomic operational tool does not just help to prevent operator faults. Every beam interruption is handled by the operator in three phases: first, he has to diagnose the fault; then he takes care that anything preventing him from making beam gets repaired and finally the operator has to recover the beam. The actual repair is often not in the responsibility of the operator, only the delegation of the repair. But as well for the proper diagnosis before as for the beam recovery after the repair he is in charge. He'll depend on his operational tools to solve these tasks quickly and efficient. If the actual repair time is on average in the same order as the time required for failure diagnosis and beam recovery, then the quality of the operational tools for these two tasks can have a significant impact on the operation statistics.

THEORY

Human-Computer Interaction

Three aspects needs to be understood for the design of good user interfaces. They are shown in Fig. 2: the goals to be achieved (the tasks), the user that has these goals (the operator) and the technology to build the user interface (the control system).

A thorough understanding of the user of operational tools, the operator, is vital to create ergonomic tools for the accelerator operation. Unfortunately many developers of operational tools are not aware that this knowledge is important. Therefore this should be the main focus of this contribution.

In some cases the operation of an accelerator facility suffers from an inappropriate specification of the tasks of the operators. The operational tools are often developed for the

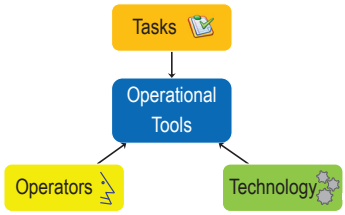


Figure 2: Tasks, operator and control system technology need to be understood for the design of operational tools.

commissioning of the facility, and therefore are optimised for the flexibility required for this phase. The tasks of the operator for the operation of an accelerator facility can differ significantly from those required for the commissioning. Problems of this type are often solved by developing new applications specifically for operation later on. A formal task analysis can help to prioritise on the development of operational tools. The task analysis section will introduce into this technique.

The technology to develop the operational tools compromises all aspects of the user interface of the control system: from input devices like a mouse, over GUI development software to the means of visualisation. This is generally among the core competence of the application developers and rarely a problem. Therefore we will omit it here.

Interaction Design

Figure 3 shows a simple model of the interaction of the operator with his tools. The operator will act on his tools by

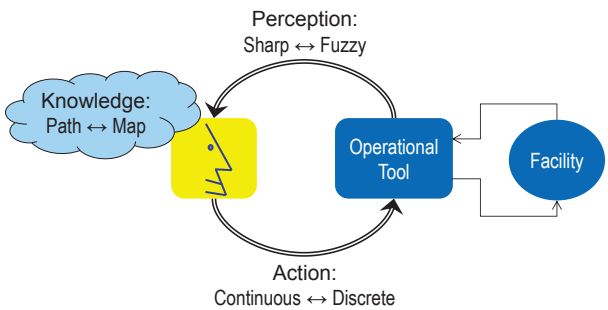


Figure 3: Simple interaction model of the operator, his operational tools and the accelerator facility.

the means of discrete control, like buttons, and continuous handles, like sliders. His actions are driven by his goal and his knowledge on how to achieve it. This knowledge can be like a path, where the operator follows step-by-step a defined procedure, or rather like a map, if the operator has a deeper understanding of his task, making him autonomous to find the way to his goal based on his knowledge. The information presented by his tool will be perceived by him either as a sharp picture, showing exactly the information he needs or rather fuzzy with room for interpretation.

A good tool should support the operator in all aspects: it should present a selection of actions appropriate to the task and easy to use by the operator, it should present all information relevant to the task in a way appropriate for the perception by the operator and it should provide all necessary knowledge required to perform the task such that they are comprehensible by the operator.

Human-computer interaction research used the theory of perception to derive guidelines for the design of effective and efficient user interfaces. The cognitive limitations of humans defining how user interfaces needs to be designed to support the user in pursuing his goals.

USER INTERFACE DESIGN GUIDELINES

While guidelines for user interfaces are formulated differently in different reference publications for user interface design, they have a great overlap in their concepts [1]. We will briefly introduce these concepts in the following, as shown in Fig. 4. Simple strategies will be provided to apply them to the design of operational tools.

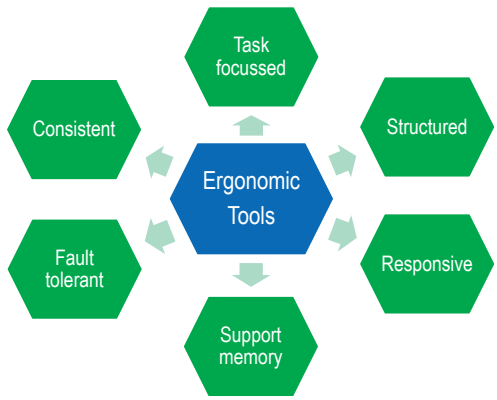


Figure 4: Design concepts for ergonomic user interfaces.

Consistency and Structure

It is obvious that consistency of the user interface has many advantages for the user. Exceptions are difficult to remember and variations in the style and layout can easily lead to confusion. Consistency is often reached without special measures, when the operational tools are created by a small number of people working closely together. If a larger number of people are involved, then a style guide helps to implement consistency in the design of the operational tools (See Fig. 5). A style guide documents a desired common style for all applications designed in a given context, like all operational tools of an accelerator facility. A variety of style aspects can be documented: the general layout of screens, the types and styles of handles and buttons to be used, the standards for the display of data, the general usage and meaning of colour and symbols or the vocabulary to be used.

The benefit of standardisation derives from the limitations of our perception. People are unlikely to mix up a picture of a close friend with the picture of a similar looking stranger, but we do confuse sometimes buttons labelled

"edit" and "exit", in particular if they are at the same location in two similar looking screens. Colour can be spotted very easily, for example a red word in a black on white text, but we have difficulties to distinguish different shades of red. We can spot relevant information quickly if it is presented in a clear, visual structure, but to find a specific word buried in a long text is very hard. The reasons for this are simple: we are good at things that helped our ancestors over the past hundred thousand years to survive. Reading text was no such thing within that time scale. And a red apple had a different shade of red anyway, depending on the light. This needs to be taken into account in the style guide: have a standardised layout and few defined handles and formats to provide structure [2]. Define few colours with specific meanings and select colours that are easy to distinguish [1]. Have a glossary of button and text labels following the simple rule: *"same name, same thing, different name, different thing"* [3]. In addition the vocabulary should be simple and unambiguous to the operators. Reading requires us to concentrate on the text, that will hinder us to concentrate on the task [1]. Use structure, colour and symbols to minimise the amount of text the operator has to read while performing his task. Many symbols have a known meaning, like a red triangle with an exclamation mark. Using them, e.g. for error messages, will catch the eye of the operator. Be careful to use "eye-catcher", like blinking text or pop-up windows: they do get the attention of the operator, but they can distract him from the task [1].

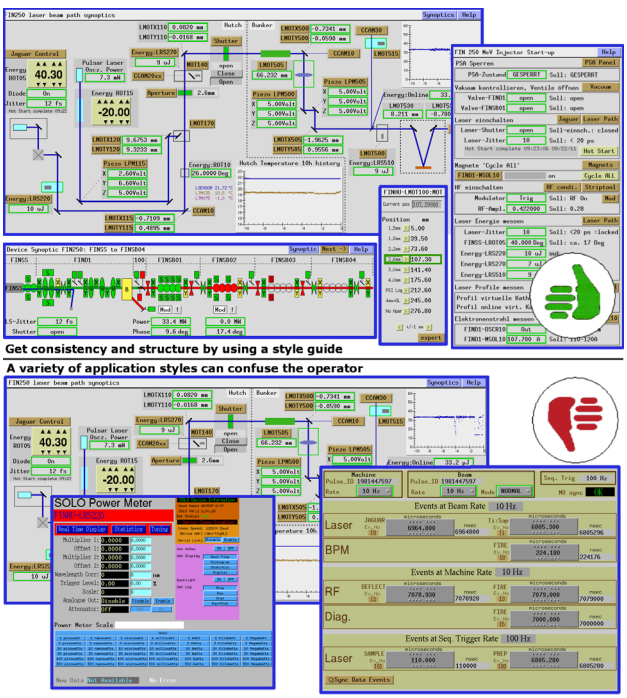


Figure 5: Use a style guide to make all operational tools similar to use. This increases the confidence of the operator in his tools and reduces the number of operator faults.

A style guide should best be ready before you start developing applications. Then it is not much extra work to adhere to it. But even after all applications have been developed, it is still a good idea to create and implement a style guide. Because many changes, like button labels or colours are very easy to change and the effort will pay off compared with the long term benefit for operations.

Task focused

A good tool should help you to achieve a goal. All your concentration should be on the task to accomplish the objective. People get easily distracted and an operator has many possible distractions: alarms to handle, telephone calls to receive, staff coming into or leaving from the control room. Therefore the tool should help the operator to stay focused on the task despite the distractions. There are several methods to help the user to focus on the task. In multi-step procedures it is useful to show which steps have been done and which still have to be done. The application should appear focused: display only actions relevant for the task and show all information useful for the task but no unrelated information.

The application itself should never distract the user from the task. Therefore it should use the task specific vocabulary, not the terminology of the application programmer. A message "division by zero, abort" will likely confuse the operator, while "the selected scan range of the measurement is zero, please select a valid scan range and retry" tells him clearly what went wrong and what he needs to do to continue his task.

Fault Tolerant

Even the best tool cannot fully prevent the operator from making errors. But ergonomic operational tools should make it hard for the operator to create significant failures and they should make it easy for him to recover from his errors. That way the operator will not just do less errors significant to operation. He'll as well feel more secure in using his tools. And his learning curve, while he's trying out new tools, will be much steeper, if he feels secure to try things out.

The confidence in his tools will even have an impact on the performance of the machine. Lets consider an SLS operator who tried to optimise the pre-accelerator with no success. He then decides to restore a saved machine setting from last week. This turns out to produce zero transmission, because - he then remembers - something had been changed yesterday on the pre-accelerator. He then learns that nobody saved the new setting. He'll be busy for an hour to restore transmission. Everyone will tell him afterwards that he should have followed the proper procedure by saving the machine settings before making many changes. This kind of stress is likely not helping him to learn the proper procedure, it is rather probable to make the operator reluctant to optimise the pre-accelerator again. It happened at the SLS, too. But we introduced an "Undo" button at the application to restore a machine setting. This allows

the operator to recover to the state before the machine settings were loaded from the file. When restoring the saved settings does not produce the desired result, he can easily get back to the previous setting. In the example he would have saved the new setting after the "Undo": this would have prevented a significant fault of the accelerator, would have reminded him of the proper procedure and would have increased his confidence in continuing the optimisation.

Responsive

Our brain has time requirements in order to perceive causality. If we click a button and nothing happens for ten seconds, then we will not be confident that things that happen afterwards are related to our actions. The following time requirements have been identified for human-computer interaction [1]:

- *0.1 sec*: immediate reaction to user input, e.g. a button appears "pressed" or a wheel-switch "turns".
- *1 sec*: status feedback if a process has not completed.
- *10 sec*: estimate time, if a process takes longer.
- *100 sec*: the typical time to make critical decisions.

A tool does not need to be fast in performing the task to meet those time requirements. It just needs to respond quickly to user input and if the task takes longer it needs to keep the user informed about the process. Human-computer interaction scientist found that responsiveness has a high impact on user satisfaction. More important for the control room is, that a responsive application clearly tells the operator what it is doing and if his attention is currently required. If it is not he can use his time for something else: fix problems to improve the performance, return phone calls to improve communication or have a cup of coffee to improve operator satisfaction.

Some examples to implement reactiveness in applications are shown in Fig. 6.

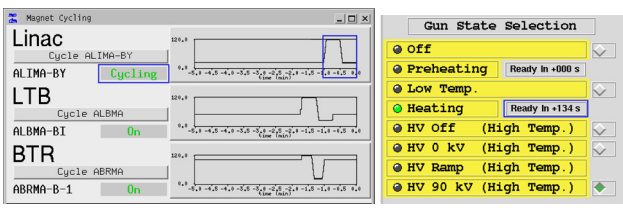


Figure 6: The magnet cycling application on the left shows the magnet current as visual feedback to the operator and the status "Cycling" while the process is still ongoing. The control panel of the electron source on the right shows a count down to forecast the end of the heating process.

Support Memory

The obvious approach to support the long term memory of an operator in performing a task is to have a thorough documentation of all the steps necessary to perform it, with a direct link from the tool to the document. Practice shows, that this works initially, most operators will read the documentation to learn the task. Some will prefer to learn the task from other operators. Once they learnt the task, only very few will read the documentation again. That should be taken into account when the documentation is updated: the operator needs to be informed about what parts have been updated and when.

There are additional methods to support the memory of the user. Each tool should clearly state it's purpose, e.g. by a descriptive title. It is often useful to provide links to related tools, to help the operator to find the tool he's looking for. Our memory works by association, therefore it will be much easier for the operator to select the proper action from a list of choices, than to recall the proper action from memory. If the number of choices is very large it is often useful to sort them by the expected frequency of usage: make those choices most visible that will be needed frequently.

A tool should provide the operator with all information required for the task. Do not expect the operator to collect the necessary information from other tools. This is particularly important during critical decision making: under stress the operator will base his decision on the available information, if the decision needs to be collected from several tools he'll may miss important information and make a bad decision.

If a tool is used to recover from a failure, then it should support documenting the failure. Do not expect the operator to type in the failure information into the electronic logbook after the failure! It is error prone and a waste of the operators time. Figure 7 shows how the magnet power supply control application at the SLS allows to save the information of the fault into the electronic logbook. This procedure is convenient for the operator but will at the same time assure that the fault is documented with complete and consistent information.

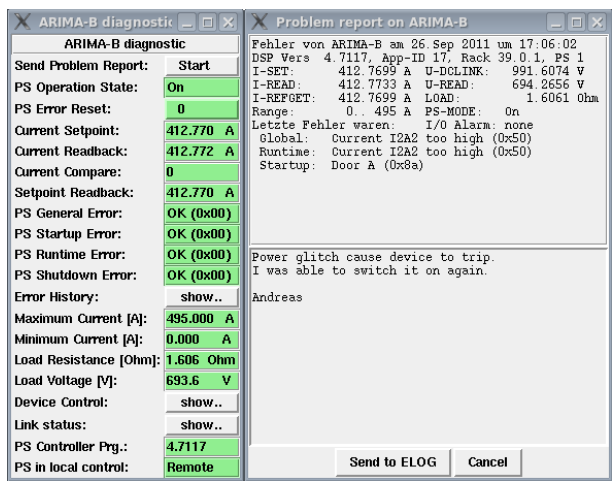


Figure 7: A problem report can be issued from the magnet power supply diagnostics application. It collects all information of the last failure and allows the operator to add a comment and save it the the electronic logbook.

TASK ANALYSIS

The method of task analysis has been developed in applied behaviour analysis and is not specific to human-computer interaction. It is the process of observing and analysing the steps of the performance of a specific task. There are different methods of task analysis and many books delve into this topic [1] [4]. We will just give a motivation here, how to use task analysis for the design of ergonomic operational tools.

Task analysis starts with observation: you watch the operators performing a task and record the steps they take to achieve their goal. The recorded steps are then organised, e.g. in the hierarchical task analysis as a tree of tasks and sub-tasks. One of the primary utilisation for task analysis is the creation of good documentation: you can use the recorded, organised steps to document the task. If you have documented the most frequent tasks of the operators, you can analyse the tools he is using for the task: Which tools are used frequently for the operator tasks? Are they appropriate for these tasks? What steps are difficult for he operator? Where are risks involved, like pressing the wrong button could cause beam interruption? Is the usage of these tools consistent or do they differ significantly?

Once you've answered these questions for the set of your operation tasks you probably know already which tools would need to be modified most urgently. But instead of starting to adapt your tools you should now start to develop a conceptual model of the operation of your accelerator. That is an idealised view on how operation should work: what tools would be useful for operation, what devices would they be used on, what are the relationships between the devices and the tools, how should they be used by the operators? Make the conceptual model as simple as possible and focused to the operator tasks, and you'll get a good starting point to prioritise your work on your operational tools.

USABILITY TESTING

Regardless on the amount of planning you've spend on the design of your operational tools: they will likely fail to meet some requirements of the users. The only way to find out is to test them on your users: the operators. Usability testing is a common method in application design [5].

A usability test will start, like a task analysis, by observing the operator using the operational tools. Here the focus is on how the operator is using the tool: Does he use it the way it has been intended? What time does he need to perform the intended task? At which point is he uncertain, where does he need guidance?

Industry builds specialised usability laboratories where hundreds of users are tested for a new product and even all eye movements of the users are recorded and analysed with special software: how long does the user look where at the screen, where does he moves the mouse pointer and when does he click where. But there is a very simple but powerful alternative for accelerator facilities with limited resources: the "think aloud" technique [2]. Tell the operator to think aloud while he's using the tool to perform the desired task. Sit behind, watch the operator and take notes. A variation of this technique is to have one operator explain to another what he's doing to perform the task. Since he is forced to explain what he is doing it will produce more information about what's going on in his mind while he's using the tool.

We use this technique frequently at the SLS. We call it "operator training": at least once a month we dedicate 90 minutes of beam time to the training of the operators. A special application called "sabotage" generates randomised system failures on demand. While one operator is responsible to fix the problem, several others are watching. His duty is to fix the problem and recover beam, to explain what he does or attempts to do and to document his actions. While the primary goal of the operator training is to exercise failure analysis and beam recovery with the operators, it serves at the same time as usability testing. Even if an experienced operator uses the tools quick and efficient, the questions from the observing operators often help to find possible improvements of the tools for novice operators.

SUMMARY

In order to design effective and efficient tools for the operation of an accelerator one needs a thorough understanding of the operational procedures. The well established user interface design guidelines of human-computer interaction theory can then be utilised to create operational tools that are intuitive and easy to use.

The optimisation of the operator interface should be an ongoing process. The techniques of task analysis can help to get a better understanding of your operational procedures. Usability testing is a proven method to improve the user interface of your operational tools. Both techniques can be implemented by regular operator training exercises on the real machine: by watching the operator actually performing his operational tasks.

The creation of ergonomic operational tools can help to achieve higher beam availability and better beam performance at all accelerator facilities.

REFERENCES

[1] J. Johnson, "Designing with the mind in mind: simple guide to understanding user interface design rules", Amsterdam, Elsevier/Morgan Kaufmann Publishers, 2010.

[2] B. Shneiderman and C. Plaisant, "Designing the user interface: strategies for effective human-computer interaction", 5th ed., Boston, Mass., Addison-Wesley, 2010.

[3] C. Jarrett, "Forms That Work: Designing Effective and Efficient User Dialogs.", Proceedings of the 50th Annual Conference of the Society of Technical Communication, Dallas, Texas, USA, 2003.

[4] J. Hackos and J. Redish, "User and Task Analysis for Interface Design", 1st ed., John Wiley & Sons, Inc., 1998.

[5] S. Krug, "Don't Make Me Think! : a common sense approach to web usability", 2nd ed., Berkeley : New Riders, 2006.

MULTI-PLATFORM SCADA GUI REGRESSION TESTING AT CERN

P.C. Burkimsher, M. Gonzalez-Berges, S. Klikovits, CERN, Geneva, Switzerland

Abstract

The JCOP Framework is a toolkit used widely at CERN for the development of industrial control systems in several domains (i.e. experiments, accelerators and technical infrastructure). The software development started 10 years ago and there is now a large base of production systems running it. For the success of the project, it was essential to formalize and automate the quality assurance process. This paper will present the overall testing strategy and will describe in detail mechanisms used for GUI testing. The choice of a commercial tool (Squish) and the architectural features making it appropriate for our multi-platform environment will be described. Practical difficulties encountered when using the tool in the CERN context are discussed as well as how these were addressed. In the light of initial experience, the test code itself has been recently reworked in object-oriented style to facilitate future maintenance and extension. The current reporting process is described, as well as future plans for easy result-to-specification linking. The paper concludes with a description of our initial steps towards incorporation of full-blown Continuous Integration (CI) support.

INTRODUCTION

Many of the systems of the CERN Large Hadron Collider (LHC) accelerator [1] and the sophisticated physics experiments placed around its circumference are controlled using commercial Supervisory Control and Data Acquisition (SCADA) technology [2]. These production control systems have an expected lifetime upwards of 20 years, nevertheless are in general implemented by an ever changing stream of developers who frequently work on many other different sub-systems too. The CERN Joint Controls Project's JCOP Framework [3] was introduced into this mix to minimize the regular hurdles faced by new developers. The JCOP Framework strives to hide much of the complexity of the underlying SCADA tool used (WinCC-OA from Siemens, previously known as PVSS [4]). The Framework further aims to provide a level of consistency in the final application, in order to minimize the cost of long term maintenance as well as for operational safety reasons. The Framework comprises not only code segments (libraries) but also a substantial amount of Graphical User Interface (GUI) [5] software. This paper outlines the deliberate steps taken to ensure that the GUI implementations are rigorously tested.

QUALITY ASSURANCE

A strict quality assurance strategy is necessary not only to deal with the long project lifetime and staff rotation issues, but also with upgrades of the JCOP Framework itself and the packages upon which it depends (e.g. operating system version, SCADA tool, compilers, database clients and servers).

GUI TESTING

Early on in the Framework project it was recognized that the GUI mechanisms comprised a significant proportion of the software being generated. Programmers are used to testing library software in an automated fashion, but GUI regression testing is often not included [6]. To have full confidence in all of the modules of the Framework package, it is necessary to automate testing of their GUI aspects also. Considerable effort has therefore been invested in GUI regression testing – on a daily basis ensuring that the individual packages and components still behave the same way they did yesterday. This approach has proved to be very fruitful, especially when new developers first start maintaining code. Inadvertent side-effects of any changes made have been identified within 24 hours.

IMPLEMENTATION

In practice, we are testing WinCC-OA panels. An initial investigation using the tool "Rational Robot" [7] was dropped when ETM [8] switched the WINCC-OA windowing system to Qt [9]. A specialised product for the automated testing of Qt GUI interfaces was selected called Squish [10], marketed by Froglogic. Like Qt itself, Squish is available on many platforms, including those used in control systems at CERN, namely Windows and Linux. Out of the box, Squish allows you to record GUI input and test the properties (including existence) of screen objects.

Squish Recording Facility

Our first implementation made extensive use of the Squish recording facility. Squish generates human readable source code that, when executed, will replay mouse gestures and keyboard input. We opted to produce Python [11] code and have grown to appreciate the power and flexibility of this language. Although Python is interpreted, for this application domain there has never been a performance issue.

In practice though, our first implementation was very hard to maintain. Our surrounding environment was in a state of constant and uncomfortably rapid flux. WINCC-OA is upgraded to a new version each year or so. Although ETM strove very hard (and were successful) in

maintaining functional backwards compatibility, their underlying implementation did change and this unfortunately was visible to Squish and the testing software. Underneath WINCC-OA, Qt itself has been undergoing a major version change also. As our test suites grew in size, more and more code needed to be changed to keep the test system working with each new version. Similar issues presented themselves with each new version of Squish. Although in general each new Squish version would bring highly desirable functional enhancements, it was usually necessary to modify and sometimes completely regenerate sections of Squish code. Our naive approach simply did not scale.

Code Refactoring

It became clear that we needed to refactor our code to make it more robust. We tackled this problem by extracting repeated sequences of recorded code into libraries, Fig. 1.

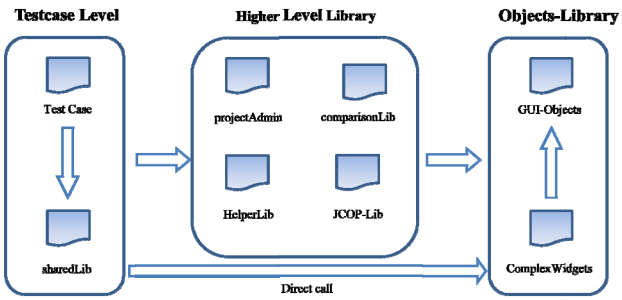


Figure 1: Library Structure.

The first step was to extract all of the routines which interact with GUI-widgets into their own library. The library was implemented in object-oriented style [12]. This programming paradigm makes it easy to define an instance of a GUI-widget once and to re-use it in several places. Our library contains several classes (one for each GUI-widget).

We then extracted all duplicate code from the test scripts into several higher-level application libraries dependent only on the above GUI-widget library. One example of an application level library would be the "Project Administration" library. This library contains a class with methods for creating, starting, stopping and deleting WINCC-OA projects, namely the functionality made available in ETM's "Project Administrator" panel. The code of these application library routines is used in nearly every test case. Extracting this code into a library makes test cases more robust and maintainable in case of changes of WINCC-OA.

The test case scripts themselves were thus reduced to library calls and verifications. Development of new tests is now quick and easy, as test creators can build on pre-existing functionality.

ENVIRONMENT

An important aspect of testing the Framework relates to the fact that the Framework runs on 2 major platforms, Windows and Linux.

One System Running On 2 Platforms

Our initial implementation attempted to validate the software on both platforms by producing a system that would execute on each of them. We opted to invoke our tests (with all of their file-system specific definitions) from within the Bash shell [13], running natively on Linux [14] but using Cygwin [15] on Windows, Fig. 2.

Figure 2: Platform commonality through Bash.

Although Cygwin is extremely powerful, the two Bashes are not identical and cause significant problems. Regularly updated versions of Cygwin provoked us into looking for an alternative solution.

One System Controlling A 2nd Platform

Squish itself is implemented as a client-server architecture, Fig. 3.

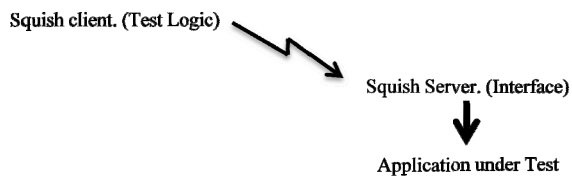


Figure 3: Squish is Client-Server.

A squish client executes the logic of the test scripts and a server executes the program being tested. This alternative arrangement was deemed to be appropriate for our multi-platform target scenario. We learned the hard way that this was not a particularly good idea. We decided that the client code would always run on Windows and would connect to the server on Windows or Linux as appropriate. Although this superficially meant that there was only one single body of testing code to maintain (always on Windows), in practice the code differed rather significantly depending on which was the target platform. As before, our problems derived from file system issues: Where was the program that was being tested going to get its data from and write its results to? These issues were not insurmountable, but they certainly made the code somewhat messy.

We encountered another much more practical problem which finally sounded the death knell to the "client-server on different machines" approach. The CERN Windows networked file-server environment was insufficiently stable to be able to work in this way. From time to time

network connections from Windows to Linux would receive a very slow response, slow enough to cause the applications and particularly their display (through the Exceed tool [16]) to decide that the network had broken. All of the windows on the target machine(s) would collapse and all of the state of the system would be lost. Any test had to be restarted from scratch – and there was no guarantee that the new attempt would not suffer a similar fate. We even tried displaying in a virtual session and viewing with VNC [17] - but this was only of partial help.

Time For A Re-think

We therefore took advantage of the code rewrite opportunity described above to revert to an architecture where the Squish client (the test logic) and the Squish server (executing the code being tested) were once again executed on the same target machine. We abandoned Bash and rewrote the test harness completely in Python, which not only made it more compatible across platforms but also made it consistent with our use of Python for the Squish test scripts.

At the beginning of the project, we were using physical Windows and Linux machines as targets for testing. We are now migrating to using Virtual Machines, hosted in the CERN Computer Centre. The advantage for us is in regard to the provision of space to house, power and cool the machines. The advantage for CERN is that our virtual machines run their jobs once per day and then go idle, releasing VM server capacity for other useful work. Thus far, our experience of this service has been excellent, with our virtual machines being functionally indistinguishable from real hardware.

The move to virtual machines has neither improved nor made worse another problem we have commonly encountered: GUI testing can be very sensitive to timing issues. Running Emu in a different environment can introduce significantly different timing behaviour. Even though Squish implements a waitForObject() function which can deal with many of the synchronisation issues, we frequently see examples where the screen update is out of step with internal data. Clicking to open up a tree structure can appear to have completed, but pressing the next mouse click too quickly can still cause problems. It is tempting to sleep for a couple of seconds or so to give the system time to catch up – but how many seconds is enough? And how many seconds is enough on all platforms, so that the code doesn't suddenly break when you change to a new operating system (Windows XP to Linux or Windows 7) or to a new software release?

ORCHESTRATING THE TESTS

At the time of the code rewrite, we stepped back, took a wider view of the system and decided to incorporate a Continuous Integration (CI) tool [18][19] that would manage execution of our tests across multiple machines, operating systems and versions. We considered using Bamboo [20] from Atlassian as a CI tool as it integrates cleanly with the other Atlassian tools (Jira, Confluence) already in use at CERN. Bamboo however needs a license and initial investigations showed it to be rather rigid in its implementation. Instead, a more configurable (and free) Open Source equivalent was selected called Hudson [21].

Hudson offers us the possibility to centrally manage the testing whilst physically distributing it to multiple machines with different operating systems and software versions installed. Tests are triggered from a central (Windows) "Hudson Master Machine" onto one or more dissimilar remote platforms, Fig. 4. The Master machine runs a web server and Hudson is configured and controlled through any browser.

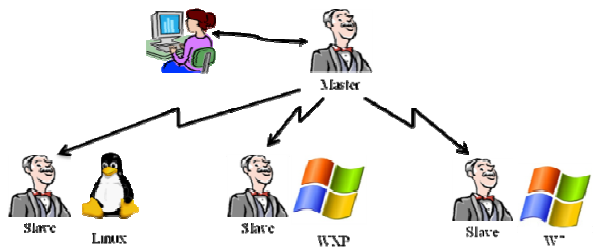


Figure 4: Hudson Master-Slave architecture.

An added bonus is that Hudson provides us with a convenient interface for having results returned to one central machine from where reports can be generated for sending to interested parties, Fig. 5.

Emu Summary (Full details here)

Topic (Test Suite)	Overall Result	OK Program Runs (Test Cases)	OK Booleans (Tests)
suite_createAVirginPvssProject	OK	1/1	0/0
suite_deleteProject	OK	1/1	3/3
suite_exerciseAnalogDigitalComponent	OK	1/1	2/2
suite_exerciseCaenComponent	OK	5/5	9/9
suite_exerciseWienerComponent	OK	3/3	6/6
suite_firstRunOfFwInstallationTool	OK	1/1	0/0
suite_installFwComponents	OK	1/1	0/0

Extract of the 14 error lines in PVSS_II.log:

Figure 5: Example result email with hyperlinks.

To date, the same central team that developed the Emu harness has also implemented the tests according to specifications drawn up with the Framework Component developers. This mode of working has several advantages:

1. The tests are prepared independently of the developers by a different individual, who is therefore unlikely to be aware of some of the implicit assumptions in the minds of those same developers. As such it is considered that these tests may be more likely to find errors than tests prepared by the developers themselves.
2. There is a written specification for the test scenarios – and consequently these are well documented. This may not always be the case in a situation where tests are written by the developers themselves.

Unfortunately this approach has a cost. The testing team becomes a bottleneck as all test code must first be written by them. For this reason we are planning to move to a scenario where the developers themselves will be able to plug their own tests into the Emu harness. Whilst alleviating the bottleneck, this comes at the cost of negating the advantages just described as well as having implications for the licensing of the software tools being used!

STATUS REVIEW

The theoretical advantages [18] of finding issues (bugs, incompatibilities, etc) before software is released, have once again been achieved in practice. New problems (side-effects) inadvertently introduced by new programmers joining the Framework development team have been spotted during the very next overnight run. We are able to test the JCOP Framework against new versions of WINCC-OA very readily by creating a new test target machine. This work is currently being extended to include the testing of a greater number of components from the JCOP Framework and also with new tests for the UNICOS Framework [22].

CONCLUSIONS

A scalable testing architecture has been defined and implemented. Automatic tests are executed every night on a variety of platforms. The architecture is not restricted to JCOP and is already being rolled out to encompass other frameworks at CERN.

AUTHOR CONTACTS

paul.burkimsher@cern.ch; manuel.gonzalez@cern.ch; stefan.klikovits@cern.ch

REFERENCES

Ask your favourite search engine for the most up-to-date links, or try the following:

[1] LHC http://public.web.cern.ch/public/en/lhc/lhc-en.html.
[2] SCADA http://en.wikipedia.org/wiki/SCADA.
[3] JCOP Framework http://j2eeps.cern.ch/wikis/display/EN/JCOP+Framework.
[4] WinCC OA (PVSS) http://www.etm.at/index_e.asp?id=2&m0id=6.
[5] GUI http://en.wikipedia.org/wiki/Graphical_user_interface.
[6] An Integration testing Facility for the CERN Accelerator Controls System, N. Stapley, M. Arruat, J.C. Bau, S. Deghaye, C. Dehavay, W. Sliwinski, M. Sobczak, ICALEPCS 2009 https://espace.cern.ch/be-dep/CO/ICALEPCS%202009/1245%20%20An%20Integration%20Testing%20Facility%20for%20the%20CERN%20Accelerator%20Controls%20System/THP085-Paper-FINAL.pdf.
[7] Rational Robot http://www-01.ibm.com/software/awdtools/tester/robot/.
[8] ETM. http://www.etm.at/.
[9] Qt. http://qt.nokia.com/products/.
[10] Squish. http://www.froglogic.com/products/index.php.
[11] Python. http://www.python.org/.
[12] Wegner, 1990; Peter Wegner, Brown University [June 1990]: “Concepts and Paradigms of Object-Oriented Programming”, Expansion of Oct 4 OOPSLA-89 Keynote Talk.
[13] Bash http://en.wikipedia.org/wiki/Bourne-again_shell.
[14] Linux http://linux.web.cern.ch/linux/scientific5/.
[15] Cygwin “http://www.cygwin.com/”.
[16] Exceed http://connectivity.opentext.com/products/exceed.aspx.
[17] VNC http://www.realvnc.com/.
[18] Continuous Integration. http://en.wikipedia.org/wiki/Continuous_integration.
[19] Fowler, 2006; Martin Fowler [01 May 2006]: “Continuous Integration”, http://martinfowler.com/articles/continuousIntegration.html#BenefitsOfContinuousIntegration.
[20] Bamboo. http://www.atlassian.com/software/bamboo/.
[21] Hudson. http://en.wikipedia.org/wiki/Hudson_(software).
[22] UNICOS http://cern.ch/wikis/display/EN/UNICOS.

ASSESSING SOFTWARE QUALITY AT EACH STEP OF ITS LIFECYCLE TO ENHANCE RELIABILITY OF CONTROL SYSTEMS

V. Hardion, A. Buteau, N. Leclercq, G. Abeillé, S. Pierre-Joseph Zéphir, S. Lê
Synchrotron Soleil, Gif/Yvette, France

Abstract

A distributed software control system aims to enhance the upgradeability and reliability by sharing responsibility between several components. The disadvantage is that it makes it harder to detect problems on a significant number of modules. With Kaizen in mind we have chosen to continuously invest in automation to obtain a complete overview of software quality despite the growth of legacy code.

The development process has already been mastered by staging each lifecycle step thanks to a continuous integration server based on JENKINS and MAVEN. We enhanced this process, focusing on 3 objectives: Automatic Test, Static Code Analysis and Post-Mortem Supervision.

Now, the build process automatically includes a test section to detect regressions, incorrect behaviour and integration incompatibility. The in-house TANGOUNIT project satisfies the difficulties of testing distributed components such as Tango Devices.

In the next step, the programming code has to pass a complete code quality check-up. The SONAR quality server has been integrated in the process, to collect each static code analysis and display the hot topics on summary web pages.

Finally, the integration of Google BREAKPAD in every TANGO Devices gives us essential statistics from crash reports and enables us to replay the crash scenarios at any time.

We have already gained greater visibility on current developments. Some concrete results will be presented including reliability enhancement, better management of subcontracted software development, quicker adoption of coding standards by new developers and understanding of impacts when moving to a new technology.

INTRODUCTION

Building, operating and maintaining a control system are complex operations. When the SOLEIL Control Group chose Tango as the distributed control system for the SOLEIL synchrotron, they firstly thought about upscaling good practices from previous laboratories to a big facility. De facto, a distributed system allows:

- Sharing between programs, as opposed to monolithic applications.
- Load-balancing to scale the number of devices to control.
- A standard communication protocol to focus development on the "business" code.

A new technology can afford the adoption of good practices only if we consider that resistance to change is an important task to manage. While we didn't call the acceptation of Tango at Soleil Kaizen, it was.

Our Kaizen : A Lean Quality

The main objectives at SOLEIL are to improve productivity and quality. Our Kaizen is inspired from the "Toyota way" or "Lean", meaning that rapid production is not possible without managing quality, and vice versa. The benefit is to detect any problems as soon as possible in the software lifecycle to reduce the cost of resolving them, which increases exponentially as production progresses.

Moreover our client has asked the Control Group to supply just-in-time solutions to their problems. So we have managed this quality project like an agile project focused on developer productivity with limited time resources.

This philosophy is based on good practices revealed by the open source community and our own experience:

- "Bottom-Up": Only the developers know how to work better. The quality process aims to generalise isolated good practices
- "Agility": We prefer modify our tools to prevent us from the non-quality instead of write documentation.
- "Don't Repeat Yourself" or DRY: We want to eliminate all small repetitive actions with no business value for the developer's job.
- "Keep It Simple, Stupid" or KISS: We won't specify our quality like a "silver bullet" with "tunnel effect" but rather by building slowly by small iterations (lasting weeks) from the actual requirement and supplying new automatism, new monitoring, new checks ASAP.
- Standardisation: Priority on components that follow our standard which can benefit from all advantages of the system. All deviations are clearly identified in our Wiki.

Only when the tools can't make a rule transparent for the developer, writing reference documentation is mandatory. But experiences show that maintenance time increases as there are many interpretations of each quality document.

Software Factory With Continuous Integration

So our quality system is mainly embodied in a Software Factory deployed in early 2008. This system integrates all tools and automation to build and monitor each piece of software. The functional perimeter includes registering new projects, building, testing and integration for deployment. The installation of the Continuous Integration principle [1] has rationalised much of the Control group's business process. This setup allowed time to be leveraged to focus on quality.

With a "just-in-time" job scheduler like Jenkins [2], the aim was to deal with automated actions at the most appropriate time. This is the case with a change in the

source code that triggers a compilation job. It's more convenient to debug code from one change than a nightly build which will compile several changes.

All our software components follow the same lifecycle defined in Figure 1, where each step is associated with a quality check.

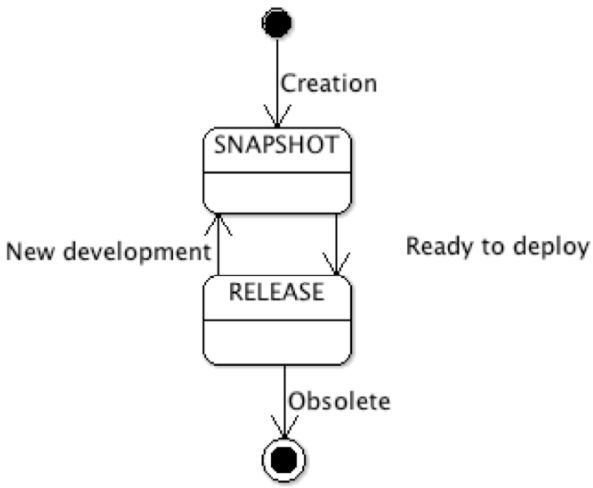


Figure 1 : Lifecycle.

In this kind of system, every new process or new tool should have the ability to be non interactive. This system is quite stable because the developers prefer to benefit from its advantages rather than use non standard tools. Even if they choose a new tool, the quality project could integrate new requirements.

The sections below will describe the extension of the Software Factory. With this new version, we chose to focus on 3 main axes: Automatic test, Continuous Quality Control and Monitoring production life.

AUTOMATIC TEST

The tests can guarantee the expected behaviour for end users as this is an important part of final quality. Although this could be the most important axis, it's also the most difficult to apply:

- No transparency: Developers have to write automatic tests themselves.
- Technically: Some difficulties with Client/Server paradigm or Graphical software
- Hardware: Equipment is often not available for automatic tests.

TangoUnit : Test For Tango Devices

TangoUnit aims to reproduce an environment for software that integrates Tango Devices. With TangoUnit, the goal is to supply a small framework to abstract registration, execution, deletion of Tango devices when the developer creates their testing environment. Tests using TangoUnit are considered as part of Integration Tests.

Simulation

An analysis made in 2010 on some beamlines reveals that only ~20% of deployed devices are directly connected to equipment. Others are process devices or from a higher layer based on the hardware equipment. It means that simulation should only consider equipment devices. This enables integration tests to be implemented for processes (high layer) devices. In the same time, some simulated devices have been created and one generic device called Transformer allows the behaviour to be changed dynamically to completely mock a real device.

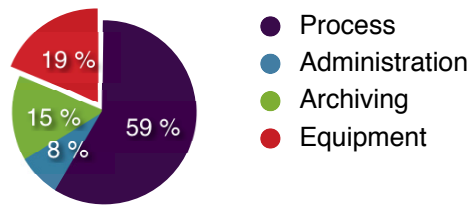


Figure 2 : Device usage.

For the ~20% of low level devices, an internal simulation mode is necessary.

Experience

How do the unit tests help to manage the recent reorganisation of projects responsibilities?

When S.Pierre-Joseph Zéphir, an ICA software engineer, took over the responsibility of the supervision project [11], she knew a little about the internal organisation of source code and the functionalities already implemented. On the other hand, the users needed to retain confidence in the one of their main tools to monitor the machine and the beamlines. If you added the current stabilisation of the new underlying graphical library and the migration of legacy components, you obtain an "explosive cocktail".

The right way to guarantee stability was to invest in unit tests especially for graphical software. Thanks to Ordinal [3], the editor of the supervision framework, which brings expertise with JFCUnit [4] a graphical testing tool, the feasibility was quickly assessed. By successive iteration, the acquired experience allowed us to better understand these black box tests on components used directly from final users.

This project cost 80% of the time spent for enhancing the tool over a period of 4 months. Initially difficult to estimate, the Return On Investment gave us more confidence in the user trust level and the comfort of portability.

A good side effect is that the rest of the team also benefited from integration tests, because the supervision software is one of highest in the dependency tree.

Next Steps

The adoption is very slow but for new project. We learned a lot from our initial success about how we can

benefit from major changes to implement tests in a project.

Today some Java projects use JUnit and a few JFCUnit for graphical tests. Some Tango Devices have a little test coverage with TangoUnit. But automatic tests are not the default practice and we will have to interest developers by training or with friendly tools.

STATIC CODE ANALYSIS

Continuous Quality Control is a process which aims to evaluate the compliance of projects to the Control Group's Quality Assurance criteria. One category specifically addresses the issue of the code quality, this is Static Code Analysis. Java developers are required to define a standard way to write code, from style to good practices.

The difficulty of obtaining a complete overview of all modules, understanding the metrics and to determining priorities from the huge metrics almost caused this project to fail.

Sonar

Sonar is an Open Source Software (OSS) developed and supported by SonarSource [5]. It aims to analyse the quality of components and report on them with a web server. The main functionality is to aggregate metrics to show only the essential data, with the possibility of monitoring metrics trends. It comes with preconfigured compliance levels for each rule.

Each new release of component triggers a complete analysis (see Lifecycle). The developer can also trigger an analysis during the development phase to allow them to anticipate the quality and correct it before the Release.

Java

Today, the Java side has been in production since late 2010 and some critical projects are actively monitored with:

- the dependency structure provides information about the abstraction level,
- the code duplication information gives maintainability level,
- the highest level violations help focusing on possible bugs.

C++

Sonar has no built-in functionality for C++. Thanks to the plugins system, it had been possible to extend Sonar. In this context, we studied the OSS market place with these constraints: multiplatform, easily parse report, no false positives and standard in the community [6]. Some tools stand out without reaching the level of Java tools.

Today the C++ analysis is integrated to Sonar through the CXX extensions co-developed by Soleil. All results come from CppCheck [7] for bug detection and Vera++ [8] for the syntax and the style of coding analysis. But we still have to invest time in determining the compliance level of each rule.

Experience

When Synchrotron Soleil decided to update the Java implementation of Tango device, G.Abeillé, the ICA engineer who has been in charge of this project, was able to check that her implementation was compliant with the OSS standard and that her unit tests were efficient.

Tango defines a standard protocol for communication between Servers and Clients. A Tango Device has some complexity with the number of execution paths with the different input and output types.

Although unit testing has a cost, we can monitor the code coverage thanks to Sonar, associated with the measure of the Cyclomatic Complexity [9]. Thus to know the effective coverage allows us to reduce the number of unit tests to the most efficient level. Others metrics like number of comments, duplication of code lines, rules compliance was useful to be OSS compliant.

Unit tests cost 2 weeks compared to the 2 months of code phase. Subsequently the unit testing cost could have been integrated in the initial time if they were written first.

Next Steps

Now the experience has benefited other projects to enhance the maintainability or recently to choose subcontractors who can comply with new ICA best practices. Sonar is also ideal to help the integration of young software developers with an accurate explanation of all rules and associated good practices. We are trying to define the monitoring process for all projects with global metrics but in an efficient way with the "Sonar Views" plugin.

POST-MORTEM MONITORING

Crash Reporting

Certainly the third success experience of this article. N.Leclercq, who is in charge of the Machine control system, enhances the quality by heading the Crash Reporting project, an "accelerators post-mortem"-like system but for Tango Control System. This process, which has been set up in production since late 2010, was motivated by the difficulty in debugging low occurrence and non-repeatable software crashes. Our accelerators operators didn't use to report these kind of events so our statistics were poor on it.

The only valuable solution was to invest in a Crash Reporting system and after evaluating the market we chose Google BreakPad [10] as the only open source and multi-OS implementation. By monitoring the current threads with another static thread able to catch all exit events, its operating principle is really non-intrusive. This library is encapsulated into the in-house CrashReport library to adjust our own parameters, such as output format or to retrieve some Tango Device informations. Then main 3rd party libraries are compiled with this CrashReport library. With the software factory, we added small pieces of code in main.cpp file of all Tango Devices used at SOLEIL. After deploying a new version of

common build parameters, the activation was carried out for all libraries and devices.

Crash Assessment is computed with each log of each Tango Device. The developer is able to replay the context thanks to the associated debugging information, although the programs are compiled with optimised options, mandatory for production deployment. Also no code instrumentation is necessary.

Quick Win

Not straightforward. This project cost approximately 80 man hours mainly due to the lack of documentation of the BreakPad project. But the ROI was immediate with working log and objective feedback each time the origin and type of failure was questioned. With this process we solved the obvious crashes caused by 30 out of the 300 Tango Devices.

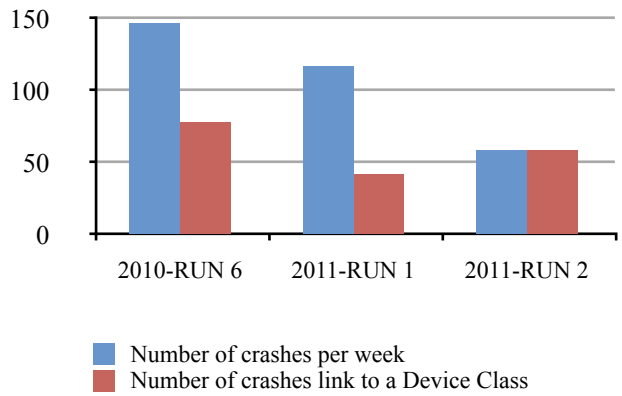


Figure 3 : Software crashes figures per Beam run (Accelerators and Beamlines).

Now we have stabilised the deviation and a maximum of crashes. Graph reports bottom out the natural law where as 80% of the crashes were caused by around 20% of the devices which corresponded with the maintenance of the global Tango Device legacy. The Crash Reporting has increased the quality of production such that the Machine's staff has clearly seen a "skyrocketing" progression.

Next Steps

This process is well monitored and maintained, but some manual operation are costly. Actually the report is made globally by hand. Other points also consist of the replay of crash context that needs the debugging info supplied with the compilation. The size of the whole distribution of binary was 5 times bigger than before. In fact, the evolution of this process will eventually be centralised like Firefox is with the Socorro project in which all crash notification will go to a server that also keeps the debugging information.

CONCLUSION

Besides of these main axis, several others monitors developed at Soleil gives us quick win results :

- Report project in activities (i.e in snapshot Status) notify developers about forgotten to release theirs projects.
- Report failed projects from dependency change is useful to analyse impact for any library evolution. This report had helped a lot to identify blocking point when we migrated all our component to the new version of Tango 7.
- Report on versions changes from the last official deployment. This report is transmitted to users to show where are the risks for the next deployment.

Obsolescence

Another point in progress we have been working on is how to target the unused Java components from the supervision software written by end-user [11]. These old components carry a lot of weight in major evolution of software. Here a simple graph analysis had allowed us to focus unit test only on used component and didn't waste our time. Cleaning the legacy should be valuable.

Human Touch

The position of the quality software manager implies good knowledge as well in software development to understand the requirement of developers than in system administration to deal with software installation. This allows to be agile with requirements.

It's important to imply the developers to these software quality processes. Sharing knowledge is also necessary for the communication and the dynamic of the team. For this purpose, we organise each month an internal regular meeting for software developers . These so-called "Café Java and C++" aims to create a dynamic for the continuous improvement of our software developments.

REFERENCES

[1] Making Continuous Integration a Reality for Control Systems on a Large Scale Basis, A. Buteau, S. Dupuy, V. Hardion, S. Le, M. Ounsy, G. Viguier, SOLEIL, Gif-sur-Yvette, ICALEPCS'09
[2] Jenkins : http://jenkins-ci.org/
[3] Ordinal : http://www.ordinal.fr/
[4] JFCUnit : http://jfcunit.sourceforge.net/
[5] Sonar : http://www.sonarsource.org/
[6] C++ Tools : http://docs.codehaus.org/display/SONAR/Cover+new+languages+with+Sonar
[7] CppCheck : http://cppcheck.sourceforge.net/
[8] Vera++ : http://www.inspirel.com/vera/
[9] Cyclomatic Complexity : http://en.wikipedia.org/wiki/Cyclomatic_complexity
[10] BreakPad : http://code.google.com/p/google-breakpad/
[11] How to Use a SCADA for High-Level Application Development on a Large-Scale Basis in a Scientific Environment, Katy Saintin, Vincent Hardion, Majid Ounsy, SOLEIL, Gif-sur-Yvette, ICALEPCS'07

SYSTEM DESIGN TOWARDS HIGHER AVAILABILITY FOR LARGE DISTRIBUTED CONTROL SYSTEMS*

S. M. Hartman†, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

Abstract

Large distributed control systems for particle accelerators present a complex system engineering challenge. The system, with its significant quantity of components and their complex interactions, must be able to support reliable accelerator operations while providing the flexibility to accommodate changing requirements. System design and architecture focused on required data flow are key to ensuring high control system availability. Using examples from the operational experience of the Spallation Neutron Source at Oak Ridge National Laboratory, recommendations will be presented for leveraging current technologies to design systems for high availability in future large scale projects.

INTRODUCTION

The Spallation Neutron Source (SNS) at Oak Ridge National Laboratory is an accelerator-driven pulsed neutron source for scientific research and industrial development. The accelerator is a high intensity H^- linac operating at 60 Hz with superconducting RF. The accelerator control system, built using the Experimental and Industrial Control System (EPICS) toolkit, consists of over 300,000 process variables on a dedicated network of approximately 1,200 nodes, including over 500 EPICS input-output controllers for controls and diagnostics, and over 100 programmable logic controllers (PLCs).

After construction, which was completed by a partnership of six US Department of Energy National Laboratories, the SNS began a simultaneous ramp up of beam power and operational hours coupled with higher availability goals. At present, the SNS has demonstrated repeatable operation at 1 megawatt of beam power (Fig. 1), and over 5000 hours of total operating time per year including over 4500 hours of neutron production time for the user program, with availability of greater then 90% (Fig. 2).

For the 2012 fiscal year, the accelerator availability goal is 90% with a 4500 hour production schedule. For the accelerator controls system, this translates to an availability requirement of approximately 99.5%

MANAGING COMPLEXITY

Large-scale, high-expense experimental science projects are increasingly undertaken as collaborative projects in response to funding pressures and system complexity. For the SNS, the partners were all US Department of Energy National Laboratories. In this case, there was a single funding agency, with all partners responsible to the same organizational and budgetary authority. For projects such as the European Spallation Source or ITER, there is the added complexity of partner contributors belonging to separate nations with responsibility for "in-kind" contributions rather then a central budget authority and organizational structure.

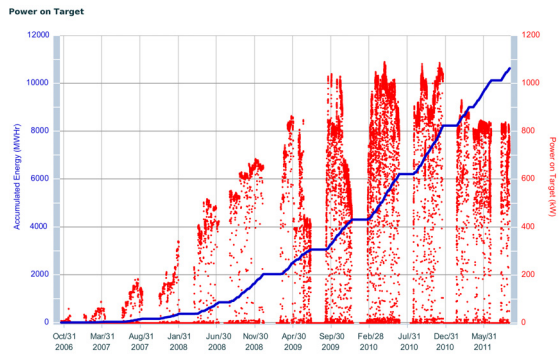


Figure 1: SNS energy and power on target, October 2006 to mid-September 2011.

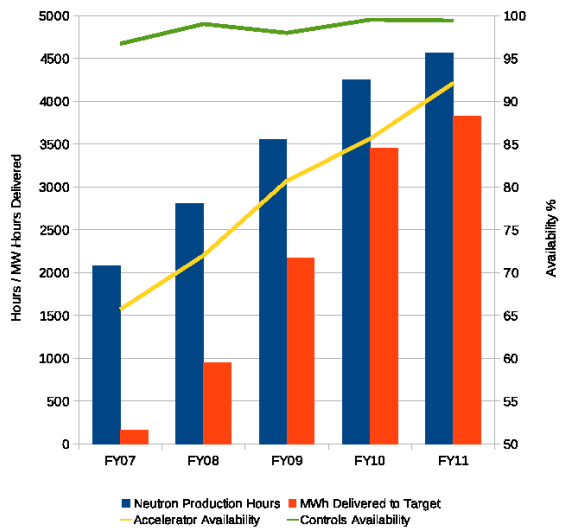


Figure 2: SNS accelerator and accelerator controls (excluding safety systems) operational performance, October 2006 to mid-September 2011.

*SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy.

†hartmansm@ornl.gov

For projects of this scope and complexity, a systems engineering approach with a holistic view of the final requirements and operational needs is critical to success. The control system serves as the interface between the various subsystems and between hardware and humans. As such, it is a key for managing this complexity. By standardizing interfaces, protocols and data handling, the control system provides the structure for bringing together the disparate parts into an overall system.

Accelerator Controls

For the SNS project, the accelerator control system group was involved early in setting standards to unify the activities of the contributing laboratories [1]. An early key decision was to standardize on a common, well developed toolkit and network protocol for use for all subsystems, namely the EPICS toolkit and the EPICS Channel Access protocol. This provided a common framework for diverse technical systems and for contributors during construction. For an operational facility, the benefit is a common set of tools and operator interfaces for accelerator controls, beam instrumentation and diagnostics, and safety systems.

Additionally, the scope for the accelerator control system was broadly conceived. The control system for conventional facility services for technical buildings, typically within the scope of the facilities or site service department, was instead included in the scope of the accelerator control system. The benefit of this decision for operations is that it becomes a simple matter to correlate, for example, the operation of a chiller or the temperature of a building with an accelerator RF structure and its impact on the beam.

Another area where the SNS control system succeeded was in the early standardization on hardware formats. Limiting the number of supported field buses and process control equipment types has resulted in simplification of long term support and maintenance. The benefit is in simpler spares management and maintenance, and fewer skills sets for which ongoing expertise is needed. Although such hardware standardization is likely to result in increased costs during the construction period, over the multi-decade life of the project, the reduced maintenance and support cost will more then offset this.

There are a few areas where early design and planning for the SNS control system has resulted in ongoing challenges for the operational period. The control system naming standard continues to be a problem. Legal names for the control system relational database are not legal in the equipment tracking and maintenance system. Names for networked devices as represented in the relational database are not compliant network host names, and it is not possible to derive the device name from a known host name. The SNS naming standard's flexibility regarding capitalization and attempts to improve readability for humans resulted in names which are neither computer parseable nor human guessable. Related to this problem is the incomplete realization of using a relational database as the definitive data source for control system tools. A lack of sufficient tools to facilitate support for early buy-in and use of the database resulted in incomplete, missing or incorrect data in the database; this still presents problems years later for control system configuration. It is critical to ensure that data entered in to the database is used from the start for actual system configuration. This ensures full investment into the system and provides the impetus to ensure data entered is correct and useful.

In recent years of the operational period for SNS, a concerted effort has been made to address control system problems that do occur at their source [2]. Review of incidents causing accelerator downtime with emphasis on identifying recurring problems and patterns has resulted in impressively high availability for the past two years. Understanding how a component, whether software or hardware, failed and how this impacted operation of accelerator systems can lead to proper fixes and long term improvements in availability.

Looking forward, an issue which needs to be addressed is obsolescence. Even though the facility has only been in operation for about five years, some custom hardware designs rely on parts which are no longer commercially available. Timing system hardware redesign is already in process [3]. At this point, industrial bus components such as VME hardware and PLCs do not appear to be a problem, or have clear upgrade paths using commercially available products. Likewise, software (such as EPICS), operating systems (Linux and VxWorks), and computing hardware (servers and workstations) have a clear path forward with reasonable compatibility available across upgrades.

The network, however, may become a problem in the future. Even after five of years of operation, the quantity of network devices on the dedicated control system network continues to grow. The initial network design resulted in a relatively flat network with a very large broadcast space for EPICS Channel Access searches. Currently, there are no significant performance issues with the network, but as the number of devices continues to increase a limit may eventually be reached. Additionally, the address space has reached a point where manual cleanup was required to allow for new devices. It would be far preferable to have the control system network more internally segmented with capacity for additional growth built in. However, such a migration would be quite disruptive at this point in the project.

Beam Line Instrument Controls

In recent months, the SNS has undertaken a review of the data acquisition and beam line controls system. Initially, this system was not included within the scope of the accelerator controls group and development took a separate path. While the accelerator control system was developed with industrial hardware (VME bus, PLCs), a mature control system toolkit (EPICS), and commercial embedded real-time (VxWorks) or open-source (Linux) operating systems,

the beam line controls were developed primarily as custom software running on commodity computers using a desktop operating system, interfacing to custom hardware. Additionally, there was less standardization on hardware, software, and system design.

Operational experience indicates a greater maintenance effort is required for this system compared to the accelerator control system. A redesign of the beam line controls system is currently in the design phase. Lessons from experience with the accelerator control system, along with beam line instrument control and data acquisition systems for other neutron sources and synchrotron sources, will be used in planning the new system (see, for example [4] [5] [6]).

The design requires a systems based approach. Beam line controls integrates a number of different subsystems including sample environment, choppers, motion control, and detector systems, and must output data in a format compatible with data analysis tools. Unlike an accelerator with a central control room and an operations staff, the experiments need to be controlled by users who may not have expertise in experiment control or data analysis. Experiments are typically scripted to allow for automated control and acquisition with minimal supervision. Nonetheless, there is significant overlap in system needs and required skill sets between accelerator controls and beam line controls.

CONCLUSION

With large user facility development projects, such as a spallation neutron source or a synchrotron light source, there is a natural progression from developing and implementing controls and diagnostics for the accelerator system to the controls and data acquisition for the experimental beam line systems. Standardization in hardware and software architecture across these systems provides numerous benefits. During the many years of project operation, the greatest expense for the controls group will be people. By sharing common technologies and skill sets for both the accelerator systems and the experimental systems, a core group can support both systems. This strategy also has the added benefit of leveling of resources between the early phases of the project focusing on accelerator development and the later phases of the project focusing on the experimental user program.

REFERENCES

[1] D. P. Gurd, "Management of a Large Distributed Control System Development Project," ICALEPCS 2001, San Jose, California (2001).

[2] S. M. Hartman, "Control System Availability for the Spallation Neutron Source," ICALEPCS 2009, Kobe Japan (2009).

[3] D. Curry, J. Dedic, X. Chen, R. Dickson, S. M. Hartman, J. Patterson, D. H. Thompson,"New Timing System Development at SNS," these proceedings.

[4] P. Lewis, *et al.*, "Data Acquisition and Instrument Control at the Lujan Center: An Update," NOBUGS 2010 Conference, Gatlinburg, Tennessee (2010).

[5] "Open GDA Project," `http://www.opengda.org/` (accessed September 2011).

[6] "Advanced Photon Source Experiment Control Prototyping," `https://confluence.aps.anl.gov/display/ExpProto/` (accessed September 2011).

THE SOFTWARE IMPROVEMENT PROCESS – TOOLS AND RULES TO ENCOURAGE QUALITY

K. Sigerud, V. Baggiolini, CERN, Geneva, Switzerland

Abstract

The Applications section of the CERN accelerator controls group has decided to apply a systematic approach to quality assurance (QA), the "Software Improvement Process", SIP. This process focuses on three areas: the development process itself, suitable QA tools, and how to practically encourage developers to do QA. For each stage of the development process we have agreed on the recommended activities and deliverables, and identified tools to automate and support the task. For example we do more code reviews. As peer reviews are resource-intensive, we only do them for complex parts of a product. As a complement, we are using static code checking tools, like FindBugs and Checkstyle. We also encourage unit testing and have agreed on a minimum level of test coverage recommended for all products, measured using Clover. Each of these tools is well integrated with our IDE (Eclipse) and give instant feedback to the developer about the quality of their code. The major challenges of SIP have been to 1) agree on common standards and configurations, for example common code formatting and Javadoc documentation guidelines, and 2) how to encourage the developers to do QA. To address the second point, we have successfully implemented 'SIP days', i.e. one day dedicated to QA work to which the whole group of developers participates, and 'Top/Flop' lists, clearly indicating the best and worst products with regards to SIP guidelines and standards, for example test coverage. This paper presents the SIP initiative in more detail, summarizing our experience since two years and our future plans.

BACKGROUND

When LHC moved from the intense preparation and commissioning phases to operations, a consequence was increased requirements on the integrity and availability from the operations crew on the released software for controls provided for by us, the Applications (AP) section of the Controls group of the Beams department. We were at the same time facing a large and ever-growing code base, demanding more and more of our time to maintain and debug with less time to focus on new functionality. Even though some quality assurance (QA) techniques, like unit testing, were being applied in several projects, no general guidelines or standards existed. Therefore, in view of improving the quality and integrity of the products released in operations, we decided to apply a systematic approach aiming to introduce quality improvement as an integral part of the development cycle and to standardize and unify between the projects with regards to deliverables and deployment and release procedures. We call this initiative SIP, the Software Improvement Process.

OBJECTIVES

The objectives set for this initiative are:

To think of and organize us as one big team, not many small ones. This means that everybody and nobody own the software produced by the section. It should therefore adhere to the standards and guidelines agreed by all of us, not follow the personal preferences of one developer. This is important in an environment where many developers collaborate on the same software and where the turnover resulting from short-term contracts are fairly high.

To achieve better quality of products that is measurable based on predefined metrics and with an agreed set of deliverables. Metrics are important as they give us the means to measure progress, which helps encouraging the developers to apply the standards and guidelines.

To reduce code base growth by promoting the development and use of common frameworks, libraries and components, avoiding duplication.

To provide better and more comprehensive documentation of the process and components.

To achieve a better software production process through incremental improvements. We don't claim to have all the answers as we start therefore we will adapt as we go and as we learn what works and what does not work, following the evolution of the industry recommendations and tools available to us.

The process focuses on three areas: The development process itself, the QA tools available to automate the process as much as possible, and how to encourage developers to include QA in their everyday work.

DEVELOPMENT PROCESS AND TOOLS

In the AP section we apply a systematic approach, a development process, to ensure a timely delivery of software corresponding to the clients needs and requests while ensuring improved productivity and software quality. It is an iterative process, where for each iteration the project goes through the stages 'Design', 'Implement, Test and Document', and 'Deploy and Maintain' as depicted in Figure 1.

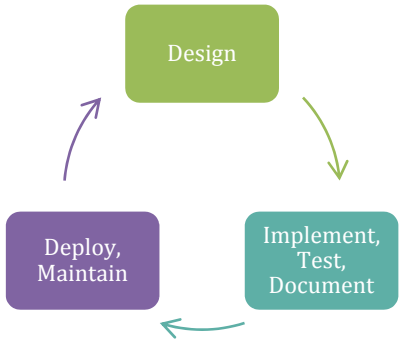


Figure 1: The stages of the development process.

For each of the stages depicted in Figure 1 we have in SIP defined the recommended or mandatory activities and deliverables. We have also identified tools that will help us automate the process as much as possible and agreed on their configuration. To ensure that each developer has these tools available and uses the same configuration, they have been integrated into an AP-specific distribution of our recommended IDE, Eclipse [1].

Design

In the design stage we have agreed to do more design reviews for new and existing projects. The purpose is twofold: firstly, it verifies the soundness of the design and propose improvements at an early stage of the iteration, i.e. before the developers invest much time in the actual coding and testing; secondly, it promotes knowledge sharing and collaboration between different development teams in the AP section, and help identify overlapping functionality, in view of reducing redundancy from our existing code base.

These reviews are at the detail level of sub-components. To discuss the design, we use UML class diagrams with the main classes and design patterns, and sequence diagrams with the interactions between these classes.

Implement, Test, and Document

In this stage the SIP focuses on three areas: the code, the documentation and the testing.

Code reviews go into more detail than design reviews. They aim at finding bugs, at making sure the code is maintainable and at verifying that our development conventions are met. However, as code reviews are very time consuming, we have decided to focus only on the most critical parts of our code (e.g. core libraries or multi-threaded code) and on code written by junior developers that need mentoring. This is done in an interactive way with person-to-person reviews of the code, but also in a lightweight, offline manner relying on the Atlassian tools FishEye+Crucible [2] integrated with Eclipse. Through this tool a committer can set up a review for a change-set, invite a number of fellow developers, which are then notified via email and can review the code changes in Eclipse and comment inline.

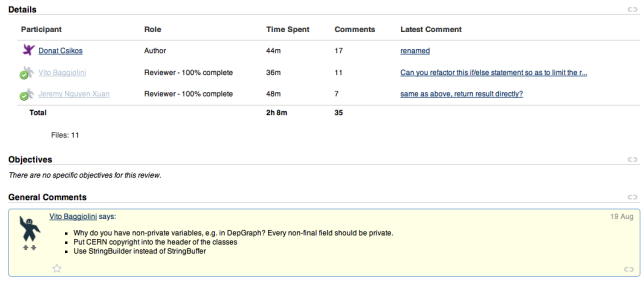


Figure 2: Code review using FishEye+Crucible.

In addition to code reviews by humans, we rely on static code analysis tools to automatically spot the most common mistakes and bug patterns. In the beginning of the SIP we performed an investigation and comparison of several tools and agreed on using FindBugs [3] and Checkstyle [4]. Plugins for these tools are distributed with our tailored Eclipse distribution, and we have agreed on a common configuration for each, also distributed with the Eclipse distribution. In addition to these external tools, we have settled on a common configuration of the Eclipse warnings, also pre-configured in our Eclipse distribution.

The benefit of having the tools integrated with the IDE is that they show up as other compilation errors, giving immediate feedback to the developer about potential problems.

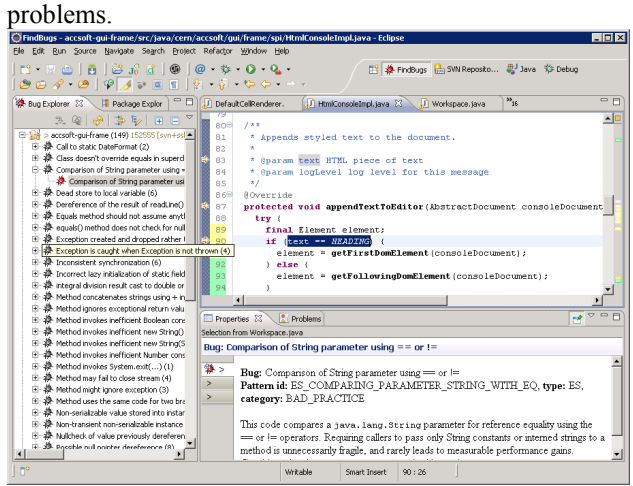


Figure 3: FindBugs report in Eclipse.

Another area the SIP focuses on is the level of testing in the projects. Even though most developers agree on the benefits of unit testing, not all take the time to implement unit tests as new functionality is added. To improve on this situation we have agreed to make unit tests a mandatory deliverable of a project: a minimum coverage of 30% for non-trivial classes must be achieved before a project can be released. We use Clover [5] from Atlassian to check the level of unit test coverage and again there is a plugin integrated with our distribution of Eclipse, giving immediate feedback to the developers of the level of coverage and the high-risk classes. Tested code appears in green, while untested code is highlighted in red (c.f. Figure 4).

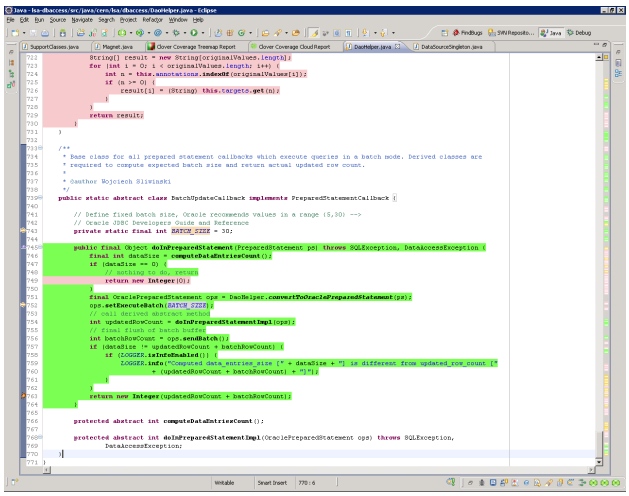


Figure 4: Test coverage indicated using Clover.

To make sure that changes in one project do not break other projects that depend on it, we have put in place a Continuous Integration (CI) server using the Atlassian tool Bamboo [6]. Whenever a developer commits changes to the source code repository, this tool checks out the new sources, compiles them and runs the unit tests. It then does the same with all dependent projects, in a cascading way, to assure that everything still compiles and all the unit tests still succeed.

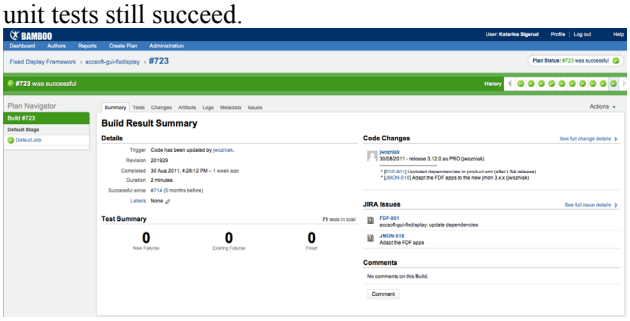


Figure 5: Bamboo build plan summary.

Documentation in SIP concerns two areas: to document the process itself and as a project artifact mandatory before a release.

To document the process, we have put in place a wiki page, detailing the set of project artifacts, best practices and standards that we have agreed on. It also lists the tools we have decided to use and their configuration.

Regarding documentation as a mandatory artifact of a project, we believe that for the documentation to be kept up to date, it should be as close to the code as possible. We therefore rely in first place on documenting the code, using the Javadoc [7] tool. At least for the base java source package(s), there should be a clear description in a file called *package-info.java,* summarizing the functionality of the package and sub-packages. As needed and for more details regarding specific sub-packages, there can be one *package-info.java* per sub-package.

Regarding documentation inside the code, we have agreed on common file and class headers, containing items like the copyright statement and SVN variables.

At least all public classes and interfaces must be documented with Javadoc following the agreed guidelines, and there should be Javadoc links to other packages (e.g. JDK).

We have also agreed on a common code formatting, available by default in our Eclipse distribution.

Both the Javadoc and the code formatting are checked using Checkstyle.

For all documentation that cannot be done using Javadoc, we are relying on Atlassian's wiki Confluence [8] to document the process and project-specific information.

Deploy and Maintain

In the deployment and maintenance phase, we have focussed on introducing a common build, release and deploy procedure using tools developed in-house. We have made a particular effort to standardize the deployment of Java server-side processes. For this we have agreed on a common naming, location and directory structure, supported by tools that enable us to easily deploy a new version of our products into operations, but also to roll-back to the previous version if necessary. The benefit is that the processes are now easily recognized and located by most members of the AP section, and allows them to intervene on processes of their colleagues, e.g. to restart or roll-back a process if necessary.

Once a process has been deployed operationally, the follow up of issues and new requests is important. We have chosen to rely on Atlassian's issue tracking tool JIRA [9] for this as it gives us the transparency we are looking for and is easily configurable to our needs. Being an Atlassian tool it also integrates well with the other tools in our development process, like the CI server Bamboo and FishEye+Crucible for code reviews where an issue number is the traceable item across all three, as shown in Figure 6.

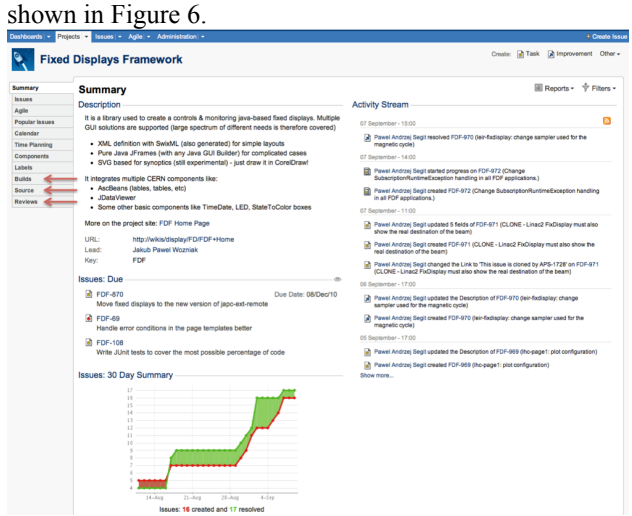


Figure 6: JIRA project summary page with links to Bamboo builds and to sources and reviews through FishEye+Crucible.

CHALLENGES

The two major challenges of SIP have been (1) to agree on the common standards and configurations, such as the common code formatting and Javadoc documentation guidelines described previously and (2) to encourage the developers to do QA.

For the first point, the approach we have taken is summarized in the first objective described above: *Think of and organize us as one big team, not many small ones*. As the code belongs to the section, not to a project or an individual developer, it should adhere to the guidelines and standards agreed by everybody not the preferences and habits of a single person.

For the second point, most members of the AP section saw the benefits of the QA techniques discussed here, and many projects already applied some of them consistently. However, they felt that their priority was to sort out issues and to provide new functionality requested by the users, and hence, they felt they did not have time for more QA work. To address this issue, we have decided to 'officialise' the time spent on QA and make the artifacts it produces mandatory deliverables of a project before release. QA objectives are part of a project's yearly objectives and prioritized and reported on regularly as with more traditional objectives. The progress for each project to adhere to the agreed guidelines is also tracked and 'Top/Flop' lists are presented on our wiki showing the best and worst projects. Finally, we have introduced "SIP days", approximately one every 3 months when everybody in the section works on a common goal, e.g. to increase the test coverage, to complete documentation or to standardize deployment configurations. This is to encourage the developers who might still have difficulties to find the time for QA activities to apply the guidelines, standards and tools in their work.

FUTURE PLANS

As for our development, we are also for the SIP applying an iterative process and continuously considering how to improve and move forward with QA related work.

Inside the section, there is still some work to finalize the list of guidelines but the main focus will be on improving the tool integration and the tracking of the progress through Top/Flop lists. As well as developing our own tool, we are also considering tools like Sonar [10], a platform to manage code quality that integrates many of the tools mentioned above, displaying the results in a clear manner.

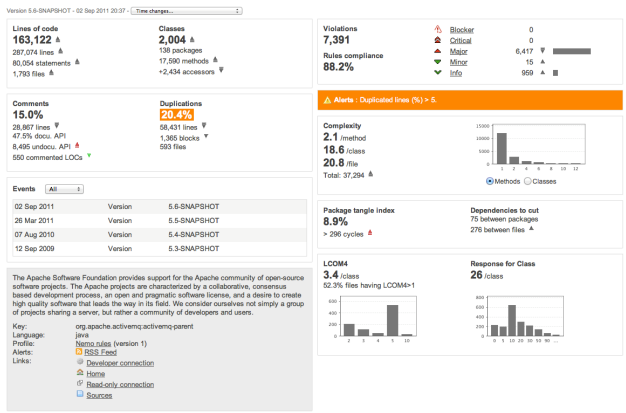


Figure 7: The Sonar dashboard.

At the group level, there is also an increased interest in QA, most particular in extending the SIP principles to C/C++ projects.

REFERENCES

[1] Eclipse: http://www.eclipse.org/.
[2] FishEye+Crucible: http://www.atlassian.com/software/fisheye, http://www.atlassian.com/software/crucible
[3] FindBugs: http://findbugs.sourceforge.net/
[4] Checkstyle: http://checkstyle.sourceforge.net/
[5] Clover: http://www.atlassian.com/software/clover/
[6] Bamboo: http://www.atlassian.com/software/bamboo
[7] Javadoc: http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html
[8] Confluence: http://www.atlassian.com/software/confluence/
[9] JIRA: http://www.atlassian.com/software/jira/
[10] Sonar: http://www.sonarsource.org/

LARGE SCALE DATA FACILITY FOR DATA INTENSIVE SYNCHROTRON BEAMLINES

R. Stotzka, W. Mexner, T. dos Santos Rolo, H. Pasic, J. van Wezel, V. Hartmann, T. Jejkal, A. Garcia, D. Haas, A. Streit, Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

ANKA is a synchrotron light source located at the Karlsruhe Institute of Technology, providing light from hard X-rays to the far-infrared for research and technology. It serves as a user facility for the national and international scientific community, currently producing 100 TB of data per year. Within the next two years a couple of additional data intensive beamlines will be operational producing up to 1.6 PB per year. These amounts of data have to be stored and provided on demand to the users.

The Large Scale Data Facility LSDF is located on the same campus as ANKA. It is a data service facility dedicated for data intensive scientific experiments. Currently, storage of 4 PB for unstructured as well as structured data and a HADOOP cluster, used as a computing resource for data intensive applications, are available. The campus experiments and the main large data producing facilities are connected via 10 GE network links. An additional 10 GE link exists to the internet. Tools for an easy and transparent access allow scientists to use the LSDF without having to bother about the internal structures and technologies. Open interfaces and APIs support a variety of access methods to the highly available services for high throughput data applications. In close cooperation with ANKA, the LSDF provides assistance to efficiently organize data and meta data structures, and develops and deploys community specific software running on the directly connected computing infrastructure.

DATA INTENSIVE SCIENCE AT ANKA

ANKA was designed and constructed by Forschungszentrum Karlsruhe in 2003 and is now operated by the Karlsruhe Institute of Technology (KIT). As a large scale research facility, ANKA provides beamtime for fundamental and application–oriented research to users from Germany, Europe and beyond [1]. Fourteen beamlines at ANKA are operational. The beamlines for analytic service cover techniques from spectroscopy to diffraction over a large spectral range from far infrared to hard X–rays. At the moment a high–resolution X–ray diffraction beamline (NANO) and an imaging beamline (IMAGE) are under commissioning and construction.

The Next Generation of Ultra Fast Tomography Beamlines

The substantial progress made in recent years, in fields like mechanics, X–ray optics and detector systems as well as the tremendous increase of processing speed and I/O bandwidth gives rise to a paradigm shift in the design and execution of tomography experiments. A new type of smart experimental station becomes possible using the vast computational power of massively parallel computation units. The process under study as well as the measurement procedure itself can be actively controlled by data driven feedback loops in direct dependence on image dynamics and spatio–temporal contrast.

Digital area detectors are an essential component for high–speed X–ray imaging experiments. A widely used X–ray detector type is the so–called indirect detector, which X–rays converts into visible light by a thin scintillating crystal. The resulting image is then magnified onto the CCD (or CMOS) camera by a diffraction limited optical system. The use of highly absorbing scintillation crystals with high light yield and optimized detector optics that are able to withstand the intense broad–band radiation produced by modern synchrotron sources has permitted a significant reduction of exposure times. High frame rates are achieved by employing column–parallel or block readout active pixel detectors, which reduce the read–out time for a full frame substantially, compared to classical high-dynamic range CCDs.

Commercially available, scientific grade cameras produce projection data at a rate of up to 750 MB/s. A detailed description of the reconstruction algorithm and its speed–up in comparison to a CPU implementation can be found in [2].

A smart experimental station for high-speed and high–throughput tomography is currently designed for the IMAGE beamline, which is constructed at the ANKA storage ring. The photon source employed for tomography will be a superconducting wiggler, which covers an energy range up to 100 keV. Compared to the bending magnet beamline Topo–Tomo [3], the flux density will be increased by more than one order of magnitude, yielding a reduction of exposure time on the same scale.

The Need for High Performance Data Management

Processing and storing the data of high–speed and high–throughput imaging beamlines are challenging tasks. Using robotic sample changers, the sample throughput is increased substantially. For instance, raw data rates presently produced at the Topo–Tomo bending magnet beam line, with a sample changing time of 15 seconds and a total scan time of 30 seconds, are 350 MB/s averaged over a daily run.

A raw reconstructed volume has a size of 50 GB, which needs to be stored alongside the raw projection data.

Data analysis is often a laborious manual process that includes several steps that cannot be automated easily, for instance finding optimal parameters of segmentation routines or even selecting pre–processing algorithms to present suitable data for post–processing. Thus, large amounts of data need to be accessible for long periods of time with high bandwidth and low latency for data processing and visualization.

The management of large scale data, its secure storage and archiving, fast and reliable access as well as high performance processing are essential for cutting edge beamline research. Usually, costs caused by growing infrastructures, reliability and sustainability of large scale data are limiting the extension of current beamline experiments.

STATE-OF-THE-ART IN DATA MANAGEMENT FOR DATA INTENSIVE BEAMLINES

The Topo–Tomo Beamline

The data management of the Topo–Tomo beamline has evolved over the years with its construction and each technical extension. Most data storage and access operations are directly steered by the user.

Samples are mounted on a rotary stage that allows a precise rotation with typical 5–10 kHz depending on the used camera. The whole detector and sample positioning setup is controlled via TANGO [4, 5] and SPEC [6, 7]. In a first step, the image data are temporarily stored on a local high performance 30 TB SAN. To check the quality of the acquired data, radiograms are monitored. Then the acquired data are processed with a dedicated four Nvidia Tesla GPU server. The final visualization for tomography is performed by VG Studio MAX [8]. It runs on a dedicated visualization server with 196 GB RAM, which allows a high performance processing of the whole tomography in memory.

Finally, the 3D tomogram plus the raw data are stored as unstructured data at the Large Scale Data Facility (LSDF). Most of the sample specific meta data that could not registered via the control application is still archived via the old fashioned laboratory book.

Data Management Systems for Beamlines

The maturity level of the data management seems to be very diverse across a variety of institutions and beamlines, and this heterogeneity is partly caused by different problems and requirements. The crucial point in beamline data management is indeed the amount of generated data and the desired throughput. In case of very data–intensive experiments the data management becomes challenging [9]. Nevertheless, beamlines with a primitive data management, where the user seems to be the main data management component and is responsible for all kinds of data storage, processing, transport and archiving, are still widespread.

An example of a data intensive beamline with complete data management during the whole data life cycle can be found at the National Ignition Facility (NIF), a research device located at the Lawrence Livermore National Laboratory [10]. The most important highlights are automatic data ingest, automatic triggering of analysis upon new data arrival, data provisioning from various data sources, policy–based automated hierarchical storage management, long term storage architecture and employment of HDF5. It is designed for beamlines with a fixed configuration that rarely needs adaptions for new scientific experiments.

Another data management system was developed and employed at SOLEIL [11]. The main idea is to produce individual data sets in form of a single and well defined atomic package containing all the relevant contextual information (including informations about the experiment itself, instrumentation, sample, user, etc.). The logical file format NeXus [12] with HDF5 as underlying physical data format is used.

The software architecture reflects the natural separation of different concerns, including hardware–layer and software–layer components for DAQ on device level, data collections and transformations and the data storage. It employs a TANGO based modular control system, so the data management components for data collection, transformation and storage, are mainly treated and integrated as TANGO devices.

Due to the required flexibility for continuously adapted beamline experiments, we think the SOLEIL Experimental Data Storage Management could be a great foundation for a sophisticated data management system, coupled with the Large Scale Data Facility infrastructure at KIT.

THE LARGE SCALE DATA FACILITY

The Large Scale Data Facility (LSDF) comprises on–line storage, data analysis computing cores, archival capacity and data management services, initially funded by the Helmholtz Association and the state of Baden–Württemberg in order to facilitate data–intensive science. Based on the experience in designing and continuously enhancing and the documented excellence of the WLCG Tier–1 centre GridKa [13] at KIT for the high-energy physics community, the LSDF is not limited to single scientific communities — instead it aims to be open for all scientific disciplines requiring large scale data storage, analysis and archiving on a regional, national and international level. Detailed plans and budget allocations exist to enhance the LSDF, in accordance with the requirements of the scientific user community.

The project has currently 4 PB of storage available, together with a compute cluster for data processing and a high speed dedicated network infrastructure, Figure 1. Access protocols, like GridFTP and the data management system iRods [14], are provided by dedicated servers attached through GPFS to the storage systems. For internal use CIFS and NFS access are offered for unstructured data.

An 10 Gigabit Ethernet network is spanned between all servers and the internet. Through dedicated high–speed network links the LSDF is connected to the research network and data sources, like ANKA inside KIT, allowing a rapid data taking. It enables scientific communities to work with and share the data in collaborations of global scale.

Based on the described infrastructure, several high level services for structured data are provided. The KIT Data Manager KDM consists of data and meta data management and repository components especially designed for large scale scientific data. The KDM DataBrowser [15] allows meta/data ingest and access using a graphical user interface that can be easily adapted for special needs. Additional services for organizing the data life cycle and data support are offered to the user communities. Open interfaces and APIs support a variety of access methods to the highly available services for high throughput data applications.

To process the experimental data a compute cluster with 60 nodes equipped with Hadoop [16], an implementation of Google's MapReduce [17] programming paradigm is directly attached to the storage. This allows for data intensive computing with excellent scalability, but requires adapted applications which run in a predefined environment. LAMBDA [18] – the LSDF Execution Framework for Data Intensive Applications — simplifies large scale data processing for scientific users by reducing complexity, responsibility and error–proneness. The description of an execution is realized as part of LAMBDA administration in the background via meta data and can be steered by the KDM DataBrowser.

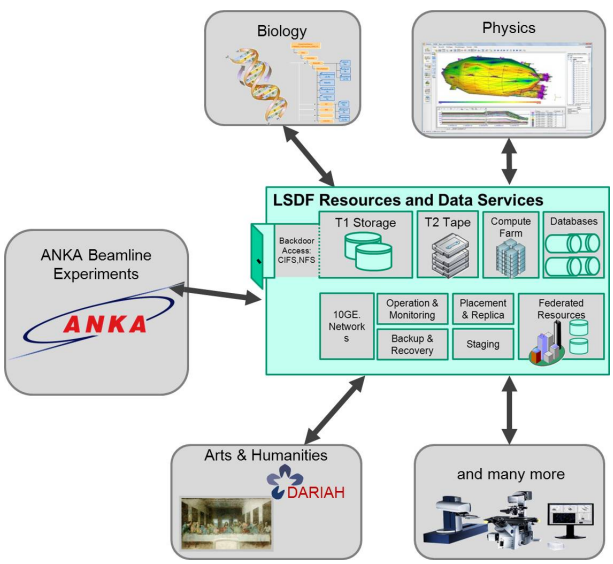


Figure 1: The Large Scale Data Facility and connected scientific communities.

If processing of experimental data requires a dedicated software environment, (for instance Windows, or specific software versions or licences) the OpenNebula [19] Cloud environment is foreseen.

In close cooperation with the user communitites, the LSDF provides assistance to efficiently organize data and meta data structures, and develops and deploys community specific software running on the directly connected computing infrastructure.

COMPONENTS OF A NOVEL DATA MANAGEMENT SYSTEM

Figure 2 depicts the conceptual data flow within a imaging beamline (left) to the storing and processing components within the LSDF. All beamline components are steered by a Control System with various user interfaces (UI). Control & monitoring data are exchanged with the components as well as user information administrated by the "User Office" for e.g. authentification and authorization.

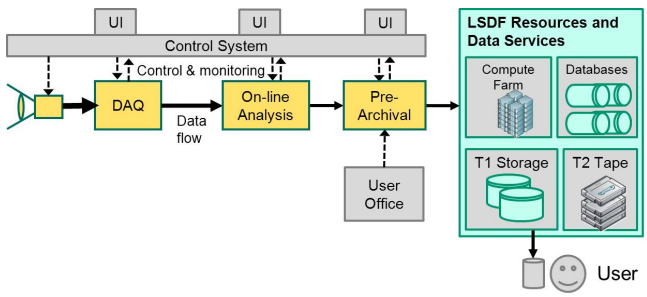


Figure 2: Data flow from the camera to the LSDF.

Starting from the detector or camera, a massive flow of data is piped to the data acquisition system "DAQ". In "DAQ" relevant measurement data are selected and annotated for further processing and storage.

"On–line Analysis" processes the data in various preprocessing steps, reconstructs tomographic images and volumes using a GPU–accelerated system and visualizes the results. Based on the visualization, the user will select promising data sets for further processing and archiving.

In the "Pre–Archival" step the date is prepared by adding meta-data for the storage, archival and processing in the Large Scale Data Facility LSDF. Additional meta data about the experiment and user data will be added from the "User Office".

Within the LSDF the data will be stored, processed and archived. The data ingest process is mostly standardized, including data transfer and meta data storage, via APIs or the KDM DataBrowser. Since the LSDF has also provided computing resources, a variety of sophisticated and computationally expensive processing algorithms can be applied automatically. For tomography beamlines algebraic reconstruction will provide high–quality images and volumes for further scientific analysis. The results will be enriched with additional meta data for data provenance and stored with the corresponding measurement data sets. Users will access their data sets on the LSDF directly via the KDM DataBrowser or a Web interface. External users will be provided with a temporary user account, according

to ANKA's policies.

It is planned to use NeXus as a primary file format for all scientific data sets written to persistent storage. The CDM technologies [20] will provide an abstract interface for accessing NeXus.

The whole architecture is designed to fulfill the requirements for continuously adapted experiments and enhanced hardware components.

CONCLUSION

The ANKA Synchrotron facility is part of KIT, a world-leading engineering institution with top ranking Faculties of Informatics, Electrical Engineering and Information Technology, and Mechanical Engineering. KIT supports a broad research palette ranging from astro–particle physics over biology to supercomputing with superior infrastructures. The modern data storage and processing infrastructures enable cutting edge research with high measurement data rates. Because the infrastructure is shared among several institutes investments are also shared. This takes some pressure off the financial budgets of ANKA and the beamline experiments.

Very economically the large scale data facility offers nearly unlimited storage, cluster computing, high throughput networks, transparent yet secure access services, and is still easily adapted to the changing requirements of the users. The unique possibilities of this facility for data intensive computing opens a new dimension for data intensive beam-ine experiments and subsequent data analysis.

ACKNOWLEDGEMENTS

Many thanks to the fruitful discussions within the High Data Rate Initiative HDRI of the German Helmholtz Association research program PNI, and with Alain Buteau and his crew from SOLEIL. Special thanks Francesca Rindone for revising and improving the texts.

REFERENCES

[1] Y.L. Mathis, B. Gasharova, and D. Moss. Terahertz radiation at anka, the new synchrotron light source in karlsruhe. *Journal of Biological Physics*, 29(2):313–318, 2003.

[2] S. Chilingaryan, A. Mirone, A. Hammersley, C. Ferrero, L. Helfen, A. Kopmann, T. dos Santos Rolo, and P. Vagovic. A gpu-based architecture for real-time data assessment at synchrotron experiments. *Nuclear Science, IEEE Transactions on*, (99):1–1.

[3] A. Rack, T. Weitkamp, S. Bauer Trabelsi, P. Modregger, A. Cecilia, T. dos Santos Rolo, T. Rack, D. Haas, R. Simon, R. Heldele, et al. The micro-imaging station of the topotomo beamline at the anka synchrotron light source. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 267(11):1978–1988, 2009.

[4] J-M. Chaize, A. Goetz, W-D. Klotz, J. Meyer, M. Perez, and E. Taurel. TANGO — an object oriented control system based on corba. In *International Conference on Accelerator and Large Experimental Physics Control Systems*, pages 475 – 479, 1999.

[5] T. Spangenberg, K. Cerff, and W. Mexner. Macro package based enhancement of SPEC controlled experimental setups. In *PCaPAC*, 2010.

[6] SPEC, http://www.certif.com/.

[7] T. Spangenberg, K. Cerff, W. Mexner, and W. Kaiser. MTANGO integration of a SIMATIC WinCC open architecture SCADA system at ANKA. In *ICALEPCS*, 2011.

[8] C. Reinhart. Industrial computer tomography — a universal inspection tool. In *17th World Conference on Nondestructive Testing*, 2008.

[9] S. Esenov, K. Wrona, and C. Youngman. European xfel daq and dm computing technical design report. Technical report, European XFEL facility, 2009.

[10] T. M. Frazier P. Adams, S. G. Azevedo, R. G. Beeler, R. C. Bettenhausen, R. W. Carey, C. B. Foxworthy, M. Hutton, L. J. Lagin, and S. L. Townsend. The national ignition facility data repository. In *Proceedings of ICALEPCS2009, Kobe, Japan*, 2009.

[11] S. G. Azevedo, R. G. Beeler, R. C. Bettenhausen, E. J. Bond, P. W. Edwards, S. M. Glenn, J. A. Liebman, J. D. Tappero, and W. H. Williams A. L. Warrick. Experimental data storage management in nexus format at synchrotron soleil. In *Proceedings of ICALEPCS2009, Kobe, Japan*, pages 75–77, 2009.

[12] Nexus, http://www.nexusformat.org.

[13] H. Marten and T. Koenig. ITIL and grid services at gridka. *Journal of Physics: Conference Series*, 219(6):062018, 2010.

[14] M. Hedges, A. Hasan, and T. Blanke. Management and preservation of research data with irods. In *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience*, pages 17–22. ACM, 2007.

[15] R. Stotzka, V. Hartmann, T. Jejkal, M. Sutter, J. van Wezel, M. Hardt, A. Garcia, R. Kupsch, and S. Bourov. Perspective of the large scale data facility (lsdf) supporting nuclear fusion applications. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, pages 373–379. IEEE, 2011.

[16] The Apache Software Foundation. Welcome to Apache Hadoop!, July 2010.

[17] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[18] T. Jejkal, J. Otte, A. Garcia, V. Hartmann, R. Stotzka, J. van Wezel, and A. Streit. Lambda – the lsdf execution framework for data intensive applications. In *Parallel, Distributed and Network-Based Processing (PDP), 20th Euromicro International Conference on*. IEEE, 2012. submitted.

[19] B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *Internet Computing, IEEE*, 13(5):14–22, 2009.

[20] Common data model, http://www.docstoc.com/docs/72729634/soleilcommondatamodelapi_1_.

COMMON DATA MODEL ACCESS; A UNIFIED LAYER TO ACCESS DATA FROM DATA ANALYSIS POINT OF VIEW

S. Poirier, A. Buteau, M. Ounsy, C. Rodriguez,
Synchrotron SOLEIL[1], France
N. Hauser, T. Lam, N. Xiong, ANSTO[2], Australia

Abstract

For almost 20 years, the scientific community of neutron and synchrotron institutes have been dreaming of a common data format for exchanging experimental results and applications for reducing and analyzing the data. Using HDF5 as a data container has become the standard in many facilities. The big issue is the standardization of the data organization (schema) within the HDF5 container. By introducing a new level of indirection for data access, the CommonDataModelAccess (CDMA) framework proposes a solution and allows separation of responsibilities between data reduction developers and the institute. Data reduction developers are responsible for data reduction code; the institute provides a plug-in to access the data.

The CDMA is a core API that accesses data through a data format plug-in mechanism and scientific application definitions (sets of keywords) coming from a consensus between scientists and institutes. Using a innovative "mapping" system between application definitions and physical data organizations, the CDMA allows data reduction application development independent of the data file container AND schema. Each institute develops a data access plug-in for its own data file formats along with the mapping between application definitions and its data files. Thus data reduction applications can be developed from a strictly scientific point of view and are immediately able to process data acquired from several institutes.

THE GENESIS OF THE PROJECT

Working independently, ESRF, SOLEIL, DESY and ANSTO software development has focused on the design of frameworks for data processing, operating on top of a NeXus [1] data storage layer.

The central issue for collaboration was related to using the same tool independent of the data container and schema. Work at ANSTO on the GumTree Data Model [2] abstracted data file access of the underlying NeXus files (the standard data format at ANSTO) by designing a data model with a set of Java interfaces. This seemed to be a very promising development to share.

SOLEIL became interested in the concept as it was coincidentally looking for a unified data access layer based on NeXus (the standard data format used at SOLEIL) to build on top of it its COMETE [3] project, a Java framework that aims to ease data visualization and data analysis applications programming.

The collaboration started between ANSTO and SOLEIL in January 2010, after a meeting between the authors at ICALEPCS 2009. The work started from the data access layer of ANSTO's GumTree project [4], written in Java.

In Q4 2011, DESY will join the collaboration and help us developing the C++ port of CDMA.

BACK TO OUR MOTIVATIONS

The important thing we must understand is that the data format is not an issue. The issue is how to allow users of our institutes to use the same tools regardless the origin of the data?

We notice that even in the same synchrotron or neutron facility, data schemas are not the same across the beamlines. The NeXus standard is useful, but not strong enough to ensure uniformity. Interpretation of the standard allows two identical beamlines to have different data schemas.

The aim of the CDMA is to offer an abstract data access layer in order to build analysis/reduction applications regardless of the data schema.

HDF-like formats allow the recording of any kind of data using an API; the physical file organization is abstracted. The NeXus-like specification is a set of logical data organizations in a standardization effort. This kind of standardization may be applicable to various tree-oriented data formats (such as HDF or XML). The problem of this approach is that facilities must produce data files that strictly comply with the specification. This is a huge challenge because each facility's staff (mostly scientists & engineers) has its own view on acquisition systems and experimental and contextual data. Also the hardware and the acquisition process are rarely driven by the data recording system.

The idea behind the CDMA is to reverse the data point of view. Rather than desperately try to standardize the data files across institutes, is it not easier to introduce a layer able to hide the different way the data is stored? We think the answer is *yes*!

We encourage the use of the NeXus standard. We believe that in the future, as the standard matures, the NeXus API may replace CDMA. We encourage facilities to take part in the NeXus standardization process. However, CDMA allows now to deal with the various standard interpretation.

1. http://www.synchrotron-soleil.fr
2. http://www.ansto.gov.au/

Each institute will continue to produce data using the most suitable format. There is no need to wait for the ultimate data organization specification before running acquisitions, and sharing data and applications!

INTRODUCTION TO THE CDMA

The CDMA comprises

- a client layer API for writing data reduction and analysis applications.
- a developer API to build the data access plug-in.

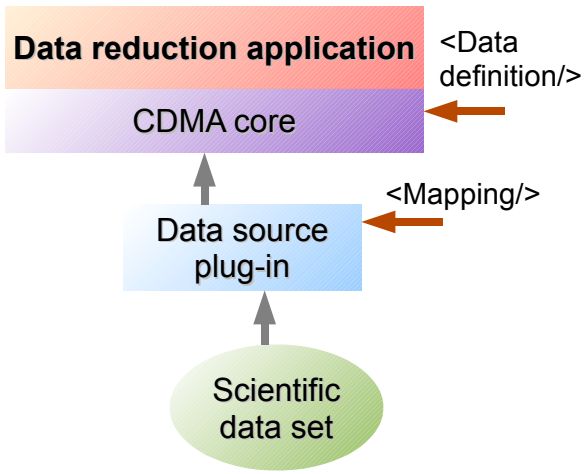


Figure 1: CDMA general usage schema.

Client API

Using the client API, the data reduction applications developer doesn't have to know anything about the file formats because the API uses an abstract data access layer that hides the data file specification.

Data Source Plug-In

The data file specification is embedded through a plug-in mechanism. It is therefore the institutes' responsibility to develop their data access plug-ins in order to open datasets acquired in their institutes through the CDMA.

Classical Navigation API

Nevertheless, using the navigation API, the data reduction developer has to know precisely the data schema in each data file their application will access. We think it's a useless waste of time. This is a roadblock to these projects being used outside of the organization where they are being developed.

Dictionary Mechanism

To solve this issue the CDMA introduces a new, innovative way of accessing data. This is the dictionary mechanism.

Using this mechanism, the data reduction application developer no longer has to know the data schema. The data is accessed using keywords. A keyword is a short character string naming a scientific measurement or a generic technical data item. Thus the data access part of data reduction applications' source code is simpler and (this is the most important) more stable.

Considering this mechanism, scientists have to agree on key names, regardless of the way the data is physically organized. Moreover, the data may have different units (wavelength vs. energy for example) for the same measurement; the CDMA provides a mechanism to perform conversions at run-time.

Please note that the old-style navigation API is still available but we strongly recommend considering the dictionary API.

CLIENT API

The client API defines interfaces that abstract the data sources. There are three levels of abstraction:

- The top level is the *IDataset* interface which represent a handle all the data of an experiment.
- *IGroup* (or *ILogicalGroup* if using the dictionary mechanism, see below) defines a group of related data. There is at least one root group for each dataset; a group may contains sub groups and data items.
- *IDataItem* defines a single value or measurement which can be a scalar or a multidimensional array.

Along with the data access layer, the Array class allows an efficient manipulating of multidimensional arrays. It is is more than a primitive Java or C++ array. It is a scientific data object allowing you to slice and dice arrays and to do math.

The CDMA provide also an class that allows error propagation, the *Error* object, that provides propagation of count uncertainties based on Poisson statistics with every math operation. This is extensible to other uncertainty calculations.

DATA SOURCE PLUG-IN SYSTEM

Using the CDMA library, the data reduction developer doesn't care about data schemas. The plug-in mechanism dynamically loads the plug-in (a dynamic library) that accesses data from the data file. Thus this mechanism allows a user to open files acquired from different institutes, in the same session.

DICTIONARY MECHANISM

The dictionary mechanism relies on two XML documents.

Data Definitions

The first document is a set a keywords matching scientific or technical data items. For instance a key named current should refer to the effective current in a storage ring at acquisition time. These keywords may be just listed by this document or organized through a tree hierarchy. In the latter case, this document describes a particular a view on the data, like a NeXus application definition.

This document is intended to be independent of the way the data is physically organized. There are (at least) two ways of writing this document:

- it may be written for a specific data analysis application which already exists and is adapted to use the CDMA,
- or it may be written independently of any application, like the NeXus application definitions.

```
<data-def name="Example">
  <group key="detectors">
    <group key="detector">
      <item key="camera"/>
      <item key="distance"/>
      <item key="exposureTime"/>
      <item key="shutterCloseDelay"/>
      <item key="xBin"/>
      <item key="yBin"/>
    </group>
  </group>
  <group key="data">
    <item key="images"/>
    <item key="spectrums"/>
  </group>
</data-def>
```

Figure 2: Data definition example.

Keywords Mapping

The second document is the dictionary itself. It's the mapping between keywords and real paths..It needs an exact knowledge of the data schema in the file.

```
<map-def name="Example" version="1.0.0">
  <item key="distance">
    <path>
      /{NXentry}/{NXinstrument}/detector/distance
    </path>
  </item>
  <item key="exposureTime">
    <path>
      /{NXentry}/{NXinstrument}/detector/Exposure
    </path>
  </item>
  <item key="shutterCloseDelay">
    <path>
      /{NXentry}/{NXinstrument}/detector/CloseDelay
    </path>
  </item>
  <item key="xBin">
    <path>
      /{NXentry}/{NXinstrument}/detector/Xbin
    </path>
  </item>
  <item key="yBin">
    <path>
      /{NXentry}/{NXinstrument}/detector/Ybin
    </path>
  </item>
<map-def>
```

Figure 3: NeXus file mapping example.

Responsibilities

Given a keywords list, the institutes that produce experimental data must write the mapping document corresponding to their data files organization.

CURRENT STATUS

There are two implementations of the CDMA, in Java and C++.

The Java implementation is mature (in operation) and already available on the SVN Codehaus repository [5].

ANSTO has developed a data browser based on the Java version of the CDMA, and uses CDMA on 4 neutron beam instruments as part of the Gumtree ecosystem.

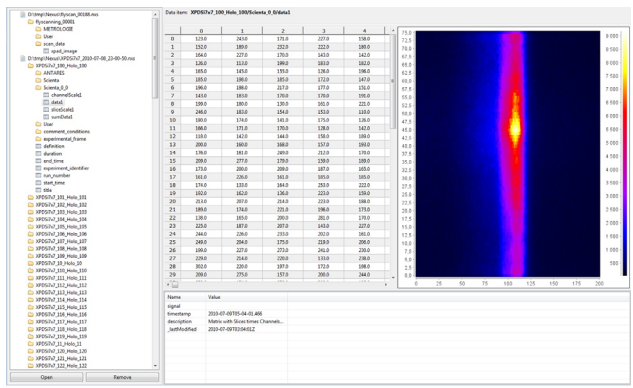


Figure 4: CDMA based data browser at ANSTO.

On SOLEIL side, two data reduction applications written in Java and based on the CDMA are currently in production, on SWING (SAX acquisitions) and ANTARES (EXAFS, Photo-emission measurements).

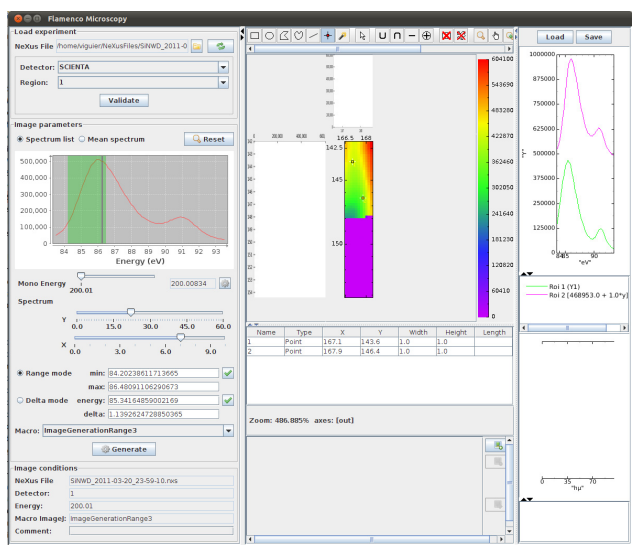


Figure 5: Flamenco, a data reduction application using CDMA at SOLEIL.

We are working on the C++ implementation which is scheduled to be available in early 2012. Then a Python port will be developed based on this C++ implementation.

CONCLUSION

The first results on using the CDMA are very promising. We have found that the time required to develop a new data reduction application has been significantly reduced, because the developers naturally no longer have to take care of the data format.

On the technical side , with the arrival of new participants (DESY, ANKA, ...), we have a community that will be able to quickly evolve this project, by enlarging the number of data reduction applications and data source plug-ins to allow cross institutes experimental files exchanges.

Last but not least, our conviction is that the CDMA project is a valuable technical answer to the Data Management issues that European projects like PANDATA, HDRI, NFFA, etc are willing to address, as CDMA provides a solution now to the old dream of able exchanging in a transparent way data files and data analysis applications between institutes.

REFERENCES

[1] NeXus format, http://www.nexusformat.org/
[2] Gumtree framework, http://gumtree.codehaus.org/
[3] COMETE framework, ICALEPCS 2011, WEMAU012 , http://comete.sourceforge.net/,
[4] Gumtree data access layer, http://gumtree.codehaus.org/GumTree+Data+Model
[5] https://svn.codehaus.org/gumtree/datamodel/trunk/

MANAGEMENT OF EXPERIMENTS AND DATA AT THE NATIONAL IGNITION FACILITY*

S. Azevedo#, R. Beeler, R. Bettenhausen, E. Bond, A. Casey, H. Chandrasekaran, C. Foxworthy, M. Hutton, J. Krammen, J. Liebman, A. Marsh, T. Pannell, D. Speck, J. Tappero, A. Warrick
Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA, 94551, U.S.A.

Abstract

The National Ignition Facility (NIF) consists of 192 pulsed laser beams fired simultaneously at a stationary target in a 10-m diameter vacuum chamber. During a typical 4-8 hour NIF shot cycle, culminating in a nano-second scale laser firing, thousands of adjustments are performed on the target (type, composition, shape), laser (beams used, their power profiles, pointing), and diagnostics (configuration, calibration, settings). It is imperative that we accurately define all equipment prior to the shot. Following the shot, and the data acquisition by the automatic control system, it is equally imperative that we archive, analyze and visualize the results within the required 30 minutes post-shot. Results must be securely stored, approved, web-visible and downloadable in order to facilitate subsequent publication. To-date NIF has successfully fired over 2,500 system shots, and thousands of test firings and dry-runs. We present an overview of the highly-flexible and scalable campaign setup and management systems that specify all aspects of the experimental NIF shot-cycle, from configuration of drive lasers all the way through archival and presentation of analyzed results.

EXPERIMENTS AT NIF

Experiments, or "shots", conducted at the National Ignition Facility (NIF) [1,2] are discrete events that occur over a very short time frame – up to 500 terawatts of instantaneous laser power are delivered to a target in tens of nanoseconds – separated by a few hours. The 192 beams of pulsed laser energy are directed at the mm-sized target centered in the NIF vacuum chamber. Each shot generates hundreds of gigabytes of data from over 30 diagnostics that measure optical, x-ray, and nuclear phenomena from the imploding target.

Most shots are part of a larger group, or campaign, of shots to advance a specific scientific goal in the understanding high-energy-density physics. One such campaign, the National Ignition Campaign, employs 1.8-Megajoules of 351-nm laser energy to implode a hydrogen-filled target as a demonstration of controlled nuclear fusion with gain. The campaign goal of fusion ignition and gain, the primary reaction that fuels stars, could potentially lead to a limitless clean energy source.

Often, in order to achieve efficient usage of precious target chamber time, shots from one campaign are interleaved with those of other campaigns. Each experiment can have very different needs from the facility including the laser controls and restrictions, target chamber controls, target positioning and cooling, industrial controls and safety subsystems.

Following each shot, the control system systematically stores the state of the entire system and all the raw diagnostic data for subsequent analysis. Extraction agents make them available outside the facility. These diagnostic data represent the principal scientific output of NIF, so a shot is not considered "complete" until they are safely archived in a secure database. Only then can the control system be reset and all subsystems can be quickly reconfigured to prepare for the next shot.

Careful planning and efficient execution of NIF shots is paramount to achieving a high volume of high quality science results. In this paper, we describe a set of integrated web-based enterprise tools that have been designed and fielded for NIF experiment support—from shot planning through presentation of final analyzed results. These tools, or follow-on designs, are expected to be employed for the projected 30-year experimental lifetime of NIF, during which many thousands of pulsed-laser shots will occur and many peta-bytes of data will be acquired and processed.

SHOT TIMELINE

NIF can fire only one experiment at a time—there is only one target, and a single chamber. For this reason, the control system [3] is designed around a "shot cycle" that executes all setup procedures for countdown to a shot as well as post-shot activities. Figure 1 shows a typical ignition shot timeline using an exponential time scale. The delivered 192-beam laser power has a very well-defined and repeatable pulse shape that is a few tens of nanoseconds in length. Prior and subsequent shots have the same tasks to perform, but their timelines are shifted by several hours. Therefore, many of the timeline activities occur simultaneously for dozens of shots, so coordination and communication among many teams are essential. Complicated campaigns can require months to years of planning for a few microseconds of total shot time, and the results are expected to be available for many years.

For the purposes of this paper, we can breakdown the pre-shot and post-shot tools into the six major categories shown in the figure and described in the following subsections—planning, setup, configure, archive, analyze, and visualize. The control system has been described elsewhere [3,4].

* This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
#azevedo3@llnl.gov

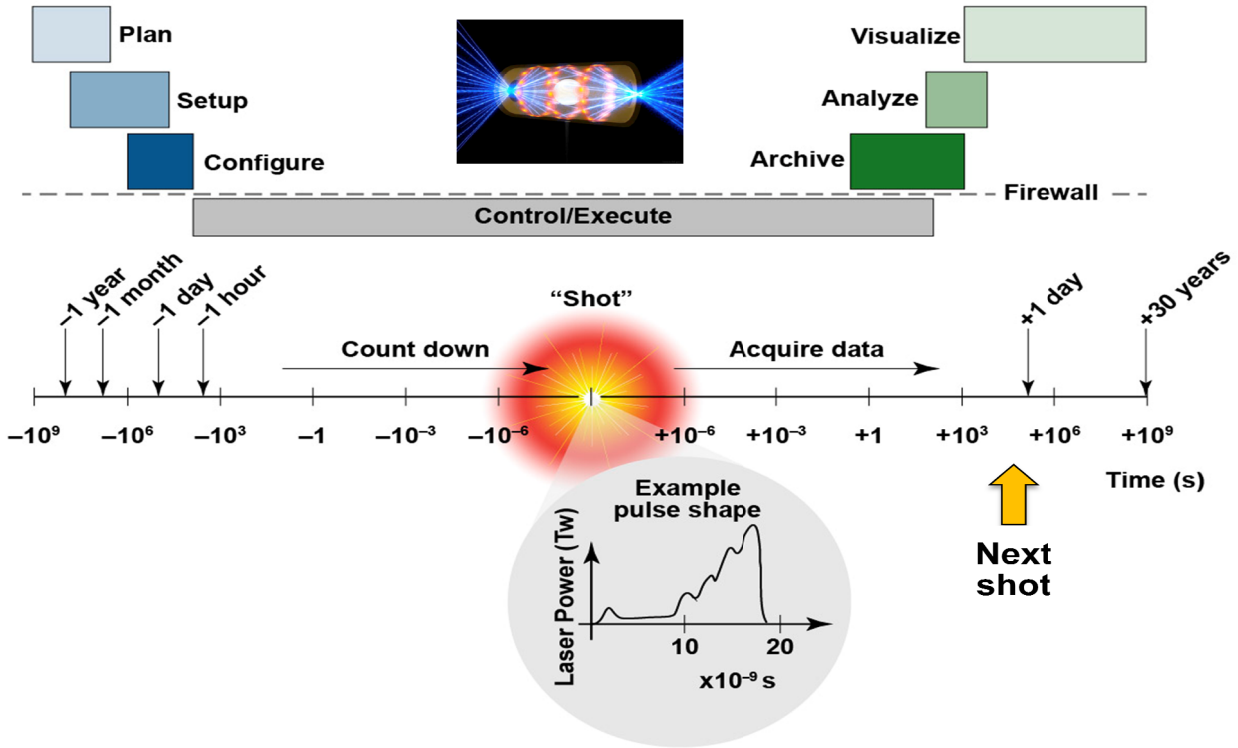


Figure 1: Shown on an exponential timeline, active control and execution of each NIF shot spans a few hours of facility time. To maximize facility usage, NIF must be rapidly reconfigured for the next shot a few hours later. Web-based tools have been developed for planning, setup, configuration, archiving, analysis and visualization of shot data.

Planning Tools

Campaign and shot planning tools provide a high-level view of upcoming experiments and facility schedules, as well as informing production and procurements. Where formerly a large shared spreadsheet was used to manage the complex decisions that dictate when and which shots will be fired, the new Shot Planning and Analysis Tool is now an on-line suite of applications that allow NIF Operations for making informed shot planning decisions based on:

- Shot dependencies, such as laser energy needed, target availability, and diagnostics required,
- Facility availability, for example, around scheduled maintenance periods or installation of new equipment,
- Predicted demand on optics, which have an expected lifetime based on cumulative laser fluence they experience, and
- Potential radiological hazards and delays caused by exposure of the facility to fusion by-products (gammas, neutrons).

Committees of NIF stakeholders use these decision support tools to evaluate the proposed shot plans, approve them and communicate that plan to the NIF community. Off-site warehouses and factories that supply the needed equipment receive advanced notice of what is needed and when. These tools help ensure that the necessary targets and diagnostics are available when needed.

The tools also enable the optimization of expensive optical consumables (lenses, mirrors, wavelength conversion crystals, etc.). NIF has an extensive optics inspection, repair and recycling program that detects and tracks micron-sized laser-induced damage sites on optics well before they can grow to impact laser performance. A rules engine is employed to predict when an optic will need replacement based on the laser energies of upcoming shots. On a single beam line, operators can intentionally block a few percent of the cross-sectional pre-amplified beam area to protect those sites, or the optic can be exchanged. In this way, the planning tools help minimize the cost of experiments by maximizing the life cycle of the optics.

Setup Tools

Once a shot or campaign is scheduled into the planning tools, a Campaign Management Tool suite (CMT) allows the experiment principle investigator (PI) to define all the necessary setup parameters for the control system to conduct the shot. This suite consists of fourteen integrated software applications that manage the preparation, validation, review, approval and configuration readiness for NIF experiments. The CMT suite manages all shot details (laser, target, diagnostics, facility), enforces the operating envelope of the laser to ensure facility safety, and provides shot setup reports and installation orders. It compares the requested facility setup to the current actual NIF configuration and notifies the PI of changes needed.

Over 15,000 setup items and over 100,000 serialized parts are monitored and controlled. The primary output of CMT is the full set of controls documentation with detailed experiment definitions to successfully perform a shot. If the entire setup documentation were to be printed as a report, it would be well over 200 pages for each shot.

Configuration Tools

When an approved shot is ready to be performed, an integrated configuration and work control suite of tools helps the on-site team ensure that NIF is ready to execute it. The tools support shot operations and data analysis by helping to enforce policies and procedures while providing a data repository to characterize the facility configuration over time. Web-based integrated applications serve to define, build, operate, maintain, and configuration-manage all components and assemblies in NIF. A transactional database monitors all installation and removal orders, while a content management database called the Location, Component and State tracking system (LoCoS) maintains the historical record of the millions of parts installed in NIF at any time.

Calibration information is also maintained in LoCoS for parts that are important to the data analysis (filters, attenuators, detectors, cameras, oscilloscopes, etc.). Re-calibration and maintenance plans are included in scheduling software. As parts are exposed to laser fluence or neutron radiation, operations personnel are notified to plan for replacement or retrofit. Work orders for part installation and removal can be automatically generated.

A NIF "Seating Chart" shows the current state of the laser, target and diagnostics on a web page that allows drill-down to every installed part and its component drawings. All drawings are under strict document control. The Shot Director will not initiate the shot cycle control system without a "green light" from the configuration tools indicating that state of the facility matches the setup request.

NIF Shot Archive

Immediately after the shot, any data potentially used in the analysis of scientific results are collected and stored in the NIF Shot Archive. This archive includes raw data and metadata, shot configuration, device calibrations and analysis parameters. In addition, the archive is also the permanent repository for experimental results produced by shots. It is meant to be the authoritative, consolidated resource for studying science at NIF. [5]

Requirements for this archive are that the data and automated results must be web-visible within a few minutes post-shot, reviewable by the scientific team for release, downloadable to authorized personnel, and secure for 30 years.

Archived data are stored in a relational Oracle Real Application Cluster (RAC) database with a software interface layer written on top of a content management framework that performs transparent object-relational mapping. This data architecture provides a powerful and flexible way that data is accessed and presented using a combination for object-relational queries or standard relational queries using SQL. Arrays of data such as waveforms and images are stored in a common Hierarchical Data Format (HDF), and are downloadable in other formats.

One key aspect to the archival of scientific data is to maintain data "pedigree" or traceability throughout all processing steps. The archive was designed to carefully maintain and validate the chain-of-custody that starts with raw data through each step of the analysis including the routines used. Pedigree is a form of metadata and is archived whenever data is refined at each step of analysis.

Analysis Tools

The Shot Analysis and Visualization (SAVI) System is an integrated set of tools for the analysis, validation and approval of science results from NIF shots. An analysis framework initiates messages to a workflow engine that launches a predefined set of analysis steps based on diagnostic type [6]. The analysis is initiated by the data as it arrives, and is performed by a highly-parallel distributed set of software components on a Linux cluster. Robust signal processing analysis tools, under version control, are customized to accommodate unique properties the each diagnostic signal or image. Results are placed back in the Shot Archive with new pedigree.

Not all analysis steps can be automated into SAVI for all diagnostics. Some, like film or trace-gas sampling, require off-line processing. Other types of new or more exotic high-speed digital systems may require specialized non-routine processing that necessitates user interaction. Also, "what-if" analysis scenarios that involve analyzing data with new, minimally-tested algorithms do not fit into the automation paradigm. In any of these cases, an alternative desktop analysis interface has been developed. Analysts can download relevant input data and pedigree, compute results using a desktop analysis code, and finally upload the results to the archive with annotations regarding their source. When both automated and desktop results appear for a particular experiment, the SAVI system provides tools to select the "authorized" results.

SAVI uses the archive to proactively validate the pedigree chain of all results in near real-time. Many different scenarios can result in a "bad" pedigree chain. For example, an improved or corrected calibration can be applied retroactively, a configuration error (e.g., wrong serial number) may have been recorded at the time of the shot, or a prior calculation step may have been re-executed whereas the final result relied on the earlier version. Any change to the quality of data inputs impact the entire pedigree chain from that point forward. Within seconds of these occurrences, the archive flags the appropriate chain of data as having bad pedigree quality, and that flag is displayed through the visualization tools. If desired, the user can then reprocess the analysis workflow with the most up-to-date inputs. Any new results created this way are stored as new versions; i.e., prior results are not overwritten.

Visualization Tools

Visualization of experimental data and results must provide a "quick look" summary of results, cross-shot analysis and trending, and collaboration tools for scientists. The web-based Archive Viewer employs rapid queries and displays to access the data in numerous ways. Additionally, collections of data can be viewed in tabular or grid formats. Hierarchical data can be traversed via hyperlinks along data relationships including between dependent results using pedigree. Scientists may download data and perform off-line desktop analysis. The application also provides an upload feature where scientists can archive desktop results to become part of the official shot record.

In 2010, the Viewer has been enhanced to include rapidly-deployable web "widgets" onto summary dashboards. Example dashboards are shown in Figures 2 and 3. Each widget loads in a separate browser thread using calls to data services hosted by visualization servers for responsive displays. Widgets in the dashboard act as windows that can be dragged, minimized, or maximized. Each widget hosts a menu of options such as data export features or actions that can performed on the data. Many of the dashboard widgets include powerful interactive data plots. Using this technique, any combination of data from the archive can be very quickly assembled into a dashboard. This method provides a current view of the data not possible with off-line tools.

The dashboard provides a collaborative environment for scientific working groups focusing on various areas of physics. Scientists view their results live (often via videoconference) from various perspective, compare results across many shots or campaigns, contrast them with simulated expectations, and select "authorized" values to be published externally.

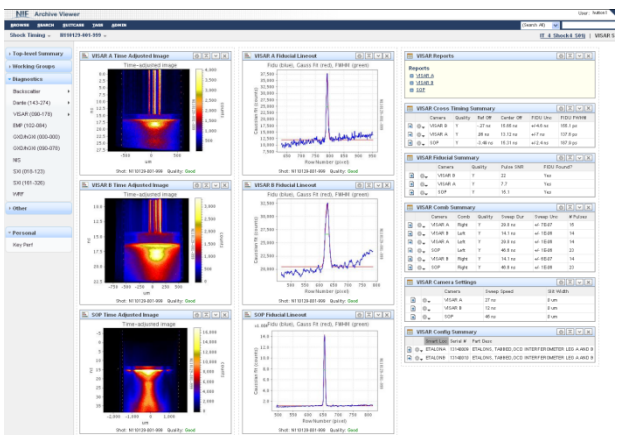


Figure 2: An example of dashboard page providing a single-shot perspective of the VISAR diagnostic.

Figure 3: An example of a multi-shot dashboard containing simulation vs. experimental data. Interactive plots allow each data series to be toggled on or off.

SUMMARY

Since the start of NIF shot operations in 2009, many hundreds of shots per year have been successfully fired. To optimize the usage of precious target chamber time, shot-based operations need flexible, high-performance tools to plan and analyze experiments. We now have over 40 integrated collaborative tools that routinely support dynamic, data-driven, high-quality shots on NIF. They also provide secure and transparent access to data, analysis and results. Shots are now being planned by many international partners, who are using these tools, for 2015 and beyond. Our future goals, as NIF transitions into a user facility in 2013, include improving the efficiency and availability of the facility. We plan to use the latest application support software to enhance the intuitive user experience, as well as to include mobile applications.

REFERENCES

[1] Moses, E.I., "The National Ignition Facility and the Path to Fusion Energy," 8th IAEA Conference on Fusion Energy, San Francisco, CA, June 2011.

[2] The National Ignition Facility official web-site, https://lasers.llnl.gov

[3] Van Arsdall, P.J., et al, "National Ignition Facility Project Completion and Control System Status," ICALEPCS '09 Conference, Kobe Japan, Oct 2009.

[4] Marshall, C.D., et al, "National Ignition Facility Control and Information System Operational Tools," ICALEPCS '09 Conference, Kobe Japan, Oct 2009.

[5] Hutton, M.S., et al., "Experiment Archive, Analysis, and Visualization at the National Ignition Facility," 8th IAEA Conference on Fusion Energy, San Francisco, CA, June 2011.

[6] Azevedo, S.G., et al, "Automated Experimental Data Analysis at the National Ignition Facility," ICALEPCS '09 Conference, Kobe Japan, Oct 2009.

LHCB ONLINE LOG ANALYSIS AND MAINTENANCE SYSTEM*

J.C. Garnier†, L. Brarda, N. Neufeld, F. Nikolaidis, CERN, Geneva, Switzerland

Abstract

History has shown, many times computer logs are the only information an administrator may have for an incident, which could be caused either by a malfunction or an attack. Due to the huge amount of logs that are produced from large-scale IT infrastructures, such as LHCb Online, critical information may be overlooked or simply be drowned in a sea of other messages. This clearly demonstrates the need for an automatic system for long-term maintenance and real time analysis of the logs. We have constructed a low cost, fault tolerant centralized logging system which is able to do in-depth analysis and cross-correlation of every log. This system is capable of handling O(10000) different log sources and numerous formats, while trying to keep the overhead as low as possible. It provides log gathering and management, Offline analysis and online analysis. We call Offline analysis the procedure of analyzing old logs for critical information, while Online analysis refer to the procedure of early alerting and reacting. The system is extensible and cooperates well with other applications such as Intrusion Detection / Prevention Systems. This paper presents the LHCb Online topology, problems we had to overcome and our solutions. Special emphasis is given to log analysis and how we use it for monitoring and how we can have uninterrupted access to the logs. We provide performance plots, code modification in well-known log tools and our experience from trying various storage strategies.

INTRODUCTION

LHCb [1] is one of the four large experiments at CERN which takes data from proton-proton collisions at the LHC. The control of the experiment and the data acquisition from the detector electronic channels to the permanent storage is managed within the Online Cluster. It is separated from the CERN IT infrastructure and managed entirely by LHCb people. The cluster is made of about 2000 Linux machines, 200 Windows machines and 60 switches and routers. Windows machines are the controllers of the detector systems. The Event-Filer farm [2], and all the other Data Acquisition (DAQ) servers are running under Linux. DAQ Software is controlled and monitored via the Farm Monitoring and Control System (FMC [3]). We use a uniform control interface for the detector and for DAQ software: the SCADA system PVSS [4].

*This research project has been supported by a Marie Curie Initial Training Network Fellowship of the European Community's Seventh Framework Programme under contract number (PITN-GA-2008-211801-ACEOLE).

† jean-christophe.garnier@cern.ch

The readout boards [5] are the interface between the detector links and the network links. An embedded computer allows the user to access and configure the entire board. It is running the Scientific Linux at CERN distribution [6].

Most of our machines are equipped with Baseboard Management Controllers (BMC) and accepts the Intelligent Platform Management Interface (IPMI [7]).

All these systems - Operating Systems, PVSS, DAQ software, IPMI, etc. - are generating numerous logs. They are traditionally consolidated locally on the system producing them. There are some limitations:

- Investigating problems can lead to correlating numerous log files from numerous locations.
- Once a machine is down, the local logs are unreachable and the problem investigation can only starts once the machine is up again.
- In case of hard drive failure, logs are lost.

We therefore implemented a central logging system, highly reliable and available, which collects logs from every system, consolidates them hierarchically and distributes them so that they can be reachable from numerous locations. Logs are then processed for indexation and analysis.

The first section presents the study and the deployment of the central logging system. The second presents the purpose of log analysis at LHCb, the tool we chose and its configuration. The third presents the status and the performance of the system, its limitations and its future.

CENTRAL LOGGING

Linux traditionally uses the *syslog* daemon [8] to manage logging from the applications and from the system. Logs are identified by a facility and a severity. The facility is usually used to define the source of the message. There are 20 facilities; hence several sources have to share the same facility in large systems. It is up to the system administrator to configure this correctly. The severity is an eight-level ladder used to classify the message importance. The daemon task is then to write different files for any combination of facility and priority. *Syslog* can be run as a client / server application. A node will send its messages to a remote node.

Syslog's features are sufficient to design a central logging system, but they were not flexible enough for us. Sharing 20 facilities amongst one node is acceptable, but not among 2200 nodes. Our system has to be able to differentiate a large number of systems, it should be able to write

several files per nodes. A few thousand files are expected. Figure 1 illustrates the file system architecture.

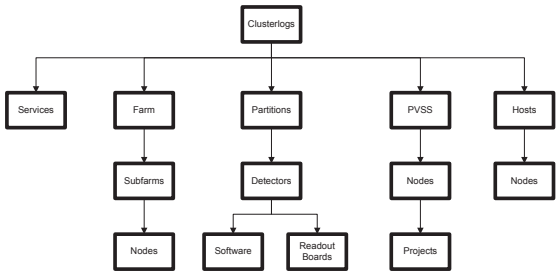


Figure 1: Log File System structure. The directory is divided first by themes. Then there are sub-categories corresponding to subsystems, then nodes, and then projects or software.

Rsyslog [9] is an enhanced *syslog* daemon. A few noticeable features are:

- A more complex analysis of the log message, using tags, regular expressions, sources, etc.
- The possibility to convert files to the *syslog* protocol, hence to get the logs from applications which are not using *syslog*.
- Multiple queues and multiple threads.
- Aggregate log lines into batches before to write them, to maximize IO performance.
- Message discard according to watermarks and log severity.

From these features, one can notice that *rsyslog* is oriented for setting up a centralized logging system. *Rsyslog* uses the *syslog* protocol so it is compatible with the *syslog* daemon. We deployed *rsyslog* on most of our cluster as client instances, except on embedded computers where it is preferable to run only *syslog*, which is much lighter.

The logging system is implemented with a few Linux servers and thousands of clients. It is described in figure 2. The log servers are configured as a cluster. The cluster nodes are identical commodity servers. Processing is not a critical issue here so they have only a single processor chip of four cores. They are equipped with 8 GB of memory. They have two SATA II hard drive disks of 2 terabytes configured as mirrors in *RAID 1*. Each node has two network interfaces. The first is used for control, for receiving the log messages and for exporting the log file system to log clients for analysis. The second is used for meta-data transfer and for cluster synchronization.

Corosync [10] manages the communication between the nodes of the cluster. On top of it, *pacemaker* [11] manages the applications that are critical for the cluster, like the various logging mechanism, virtual IP addresses, etc. It makes sure that the nodes and the services that they provide are running correctly, and distributes them amongst the cluster. Logs are written in a common area for all the cluster nodes, using *gluster* [12]. This is a file system that we use to replicate the information between all the nodes of the cluster, in order to increase the reliability of our system. Hence, if we have four machines with 2 terabytes hard disk space, our cluster area will only have a maximum of 2 terabytes space. *Rsyslog* is running as a server on every cluster nodes. This particular daemon is not managed with pacemaker as it has to run on every instance. It is simply managed via *chkconfig*. This is the main logging software, but there are a few more custom software. They are managed by pacemaker, and they process custom logging protocols developed for LHCb. The cluster is configured an Active/Active way. All nodes are processing logs coming from clients in parallel. In order to do that, we use a virtual IP address that every node recognizes as theirs. This IP address is bounded to a multicast Ethernet address that the switch uses to send data to all servers. Each server receives every log messages, but they will only process a subset of them, according to a hashing algorithm managed by *netfilter* [13, 14].

The log servers all share the same configuration for *rsyslog*. The current configuration is awfully complex, but it is interesting to stress that it currently does not rely on multi-queuing and multi-threading as we observed that performance were decreasing so much that no logs were written in the end. We suspect a deadlock or live-lock issue and further investigations are required before to report about that to the developers. The daemon is configured to write batches of several lines, in order to improve IO performance, trying to do not delay the log information too much for the user.

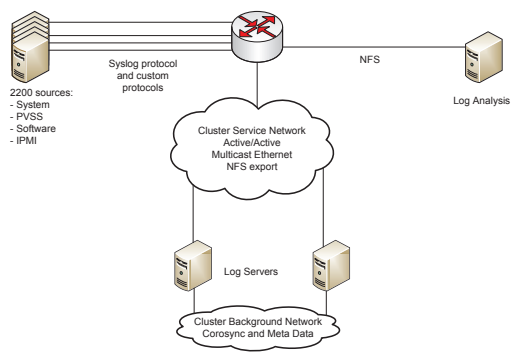


Figure 2: Architecture of the logging system. 2200 clients are logging information using the *syslog* protocol into an Active/Active cluster. This cluster exports the log files via NFS for log analysis. Log analysis is then performed by user applications and a host based intrusion detection system.

Clients are running both Windows and Linux. Under Linux, *rsyslog* sends the system logs, most application

logs, PVSS and IPMI logs to the log cluster. Our farm and our data acquisition servers are however running applications which are not logging to *syslog* but to FIFOs which are then read by FMC loggers. These loggers can then directly write a file or forward logs to a remote FMC logger. The log cluster runs FMC loggers, but some of their proprieties prevent them to run Active/Active like *rsyslog*, so pacemaker makes sure that only one server runs one instance of *rsyslog* for a same system. The event logging system of Windows is slightly different and does not rely on *syslog*. The system relies there on *Snare* [15] and *Epilog* [16] to respectively send event log messages and PVSS logs to the cluster using the *syslog* protocol.

The log cluster exports the log file system via NFS to some servers, so that users can access all logs easily. A special server is performing log analysis.

LOG ANALYSIS

Log analysis consists in reading the log files to look for patterns or sequences of lines. Most of the lines will not be interesting and nothing will be produced, in some cases a message is written in a dedicated log file which is a summary of what was important. Really important log messages will trigger alarms, sent by e-mail or text message, and they could also trigger automatic responses to fix the problem.

Several tools exist and are open source, but our choice is the Open Source Host-based Intrusion Detection System OSSEC [17]. Performing log analysis using a Host-based Intrusion Detection System (HIDS) comes along with a wider strategy of security in LHCb. It will be peered with an IDS performing network analysis.

OSSEC brings numerous features as it is a client server HIDS, but our primary use here will be to configure it as a log analyzer. It consists in defining decoders, which are templates that a log line can match or not. If a line matches a decoder, then OSSEC can apply rules to this line, in order to study it, to define its degree of importance, and to decide if the administrator should be warned, immediately or later, and whether to run or not an automatic action to fix the problem.

The OSSEC configuration is complex for our cluster, and there is still a lot of tuning to be done. In order to keep tracks of old configurations, the OSSEC rules and decoders are written into a GIT [18] repository.

RESULTS

We successfully implemented a central logging service with log analysis. It collects most of the identified logs of the LHCb Online system and writes them in more than 12000 files. Figure 3 shows the performance of a node over one month. The network is not overloaded but some peaks can make performance quite weak. The scalability, the reliability and the high availability requirements are fulfilled. There are however a lot of improvements to implement.

The typical issue of such system is the “split brain” issue. It happens when one or more members of the cluster cannot communicate with the other members for a while, and are starting to work independently from each other. Pacemaker can make sure that such situation does not happen, whereas it occurred with gluster a few times. It is not a critical issue at all there, as once the cluster is unified, gluster will recover files which were written by both sub-clusters. Figure 4 shows the IO performance of the hard drive of a node when such recovery occurs. The nodes are basically unable to write any log messages for a while as their hard drive disks are monopolized by gluster to check the 12000 files of our system and to merge data from the sub-clusters. At first the log servers were designed with 4 GB of memory and the recovery was a critical moment as all log messages are cached in memory. Doubling the memory made this situation more comfortable. If the gluster area size increases much more, this recovery might take much more time and the caching might not be enough to overcome it. In that case, upgrading the hard drive disks to formats which can support a larger number of IOs would improve the situation.

Another issue comes from the Active/Active configuration of the cluster. Netfilter is a standard module of the kernel. Its project which manages the virtual IP address and the multicast Ethernet address is however written with a hard-coded debug logging. It prints a line for each packet received. The log files for the log servers are basically polluted with these messages. There are two solutions. The first is to use another netfilter module, more recent and not integrated into pacemaker yet. The second, that we adopted, is to patch the kernel module to remove the disturbing lines.

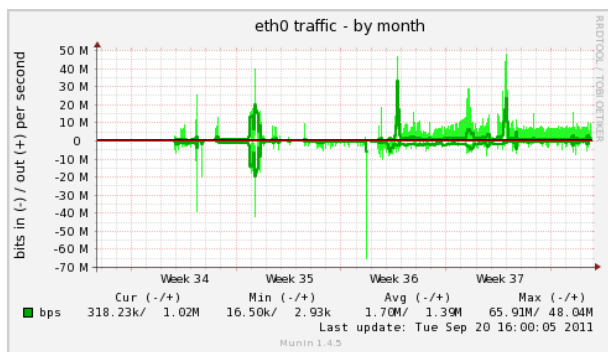


Figure 3: Log server throughput for one month. The input rate are log lines coming to *rsyslog* or custom loggers. The output rate is the NFS rate.

The system would ideally be real time, users could read logs remotely as soon as they were written locally by the nodes, and the log analysis would be performed at this moment as well. It is not possible yet for several reasons. In order to maximize IO performance, *rsyslog* is configured to write batches which produces a slight delay. Thereafter, our current system is unable to export the *gluster* file sys-

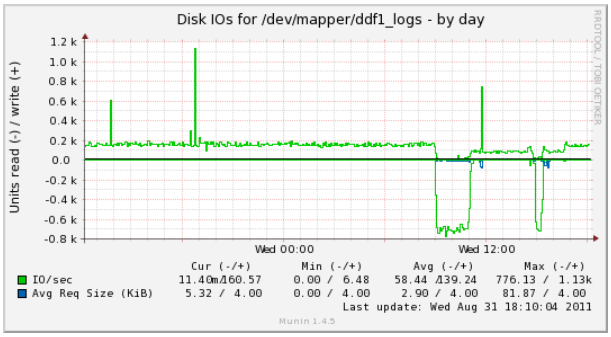


Figure 4: Disk IOs on the RAID file system. The read peaks around Wednesday 12:00 were gluster split brain recovery situations. During these periods, all IOs operations are allocated for gluster and rsyslog cannot write most of the logs. It still performs caching.

tem via NFS, so it exports the real file system. Hence, the log analyzer and the users have to wait for the *gluster* cache to be flushed to disk. An alternative could be to user a *gluster* client instead of NFS.

LHCb, and more particularly Fotis Nikolaidis, contributed to the *rsyslog* module *imfile* which manages the translation of files to syslog. With this module, *rsyslog* polls files periodically, and logs the difference between the previous poll and the current poll. The new version of the module provide the option to use *inotify* [19] under Linux instead of polling. This tool allows applications to register in order to be informed when changes happens to files. Hence, changes in the log files are directly reported to *rsyslog*.

REFERENCES

[1] The LHCb Collaboration, Augusto Alves Jr and others, "The LHCb Detector at the LHC", JINST, 3:S08005, 2008.

[2] http://lhcb-trig.web.cern.ch/lhcb-trig/HLT

[3] https://lhcbweb.bo.infn.it/twiki/bin/view.cgi/LHCbBologna/FmcLinux#Introduction

[4] http://lhcb-online.web.cern.ch/lhcb-online/ecs/PVSSIntro.htm

[5] http://lphe.epfl.ch/tell1/

[6] http://linux.web.cern.ch/linux/

[7] http://www.intel.com/design/servers/ipmi/

[8] http://datatracker.ietf.org/wg/syslog/charter/

[9] http://www.rsyslog.com/

[10] http://www.corosync.org

[11] http://www.linux-ha.org/wiki/Pacemaker

[12] http://www.gluster.org/

[13] http://www.netfilter.org/

[14] http://security.maruhn.com/iptables-tutorial/x8906.html

[15] http://www.intersectalliance.com/projects/SnareWindows/index.html

[16] http://www.intersectalliance.com/projects/EpilogWindows/index.html

[17] http://www.ossec.net/

[18] http://git-scm.com/

[19] http://lwn.net/Articles/104343/

INSTRUMENTATION OF THE CERN ACCELERATOR LOGGING SERVICE: ENSURING PERFORMANCE, SCALABILITY, MAINTENANCE AND DIAGNOSTICS

C. Roderick, R. Billen, D. Dinis Teixeira, CERN, Geneva, Switzerland

Abstract

The CERN accelerator Logging Service currently holds more than 90 terabytes of data online, and processes approximately 450 gigabytes per day, via hundreds of data loading processes and data extraction requests. This service is mission-critical for day-to-day operations, especially with respect to the tracking of live data from the LHC beam and equipment.

In order to effectively manage any service, the service provider's goals should include knowing how the underlying systems are being used, in terms of: "Who is doing what, from where, using which applications and methods, and how long each action takes".

Armed with such information, it is then possible to: analyse and tune system performance over time; plan for scalability ahead of time; assess the impact of maintenance operations and infrastructure upgrades; diagnose past, on-going, or re-occurring problems.

The Logging Service is based on Oracle DBMS and Application Servers, and Java technology, and is comprised of several layered and multi-tiered systems. These systems have all been heavily instrumented to capture data about system usage, using technologies such as JMX.

The success of the Logging Service and its proven ability to cope with ever growing demands can be directly linked to the instrumentation in place.

This paper describes the instrumentation that has been developed, and demonstrates how the instrumentation data is used to achieve the goals outlined above.

INTRODUCTION

Born out of the LHC Logging project, and operational since 2004, the CERN accelerator Logging Service (herein referred to simply as the "LS") is used to store and retrieve billions of data acquisitions per day, from across the complete CERN accelerator complex, related sub-systems, and experiments [1].

The LS is considered a mission critical service, heavily relied upon to support day-to-day operation. As such the availability and performance of this service are paramount.

ARCHITECTURE OVERVIEW

Figure 1 shows a basic overview of the LS which is essentially comprised of:

- Two Oracle databases: A so-called Measurement database (MDB) where raw data from Java processes and other Oracle databases is persisted during seven days, and a Logging database (LDB) where a sub-set of MDB data and pre-filtered data from industrial SCADA systems are stored on-line indefinitely.
- Distributed Java APIs are responsible for loading data into the databases.
- A sub-set of MDB data is transferred to the LDB using in-house developed PL/SQL code that uses a comprehensive set of metadata to dynamically filter the data of long-term interest.
- A powerful distributed Java API is the sole means of extracting data from the databases, which includes a command line interface. Applications wishing to use the API must be pre-registered. At the time of writing there are 100 applications registered to a heterogeneous client community. Direct SQL access is not permitted.
- A generic Java GUI called TIMBER is also provided as a means to visualize and extract logged data. The tool is heavily used, with more than five hundred registered users.

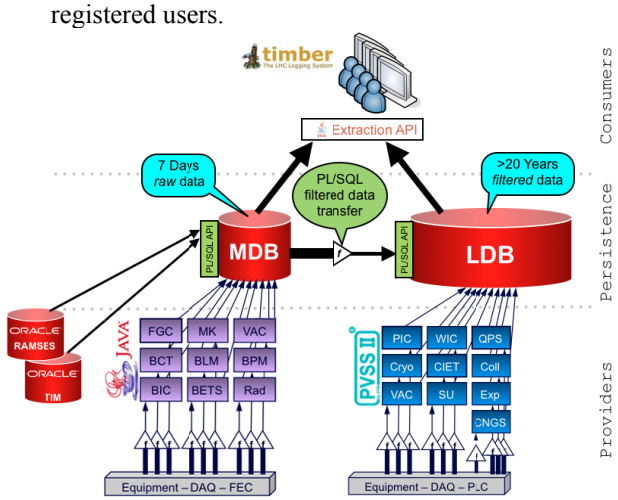


Figure 1: Logging Service architecture overview.

The Java APIs for both logging and extracting data are significantly optimized, and run on Oracle application servers. For data extraction clients, the fact that database access is actually made in a distributed manner via an application server is hidden within the Java API.

PERFORMANCE, SCALABILITY, STABILITY … AND USERS

The LS is a high performance service, which needs to deal with very high data throughputs. At the time of writing the MDB has to process approximately 5.4 billion records/day, which equates to around 270GB/day (annual throughput of approximately 100TB). Meanwhile the LDB needs to persist around 4 billion records/day for

some 850 thousand signals. This boils down to storing 140GB per day (50TB/year) and keeping it available online beyond the lifetime of the LHC.

The success of the LS leading to an increase in scope beyond the LHC, together with unforeseen events requiring more data to be available, has meant that the current data throughput levels far exceed initial expectations, which predicted 1TB/year during LHC operation.

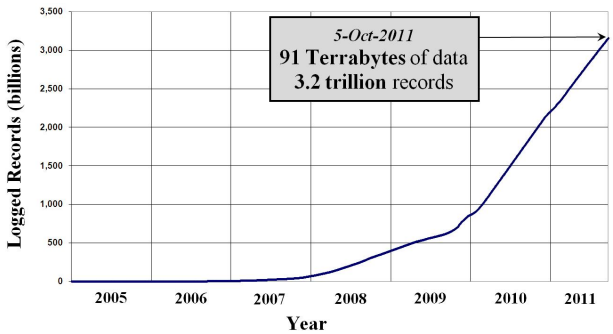


Figure 2: Evolution of logged data.

Figure 2 shows the evolution of logged data, and clearly illustrates how the LS has had to scale to satisfy evolving requirements. The ability to scale in such a manner is in no small part down to the design of the LS [2], however instrumentation also plays an important role, as will be explained later in this paper.

The amount of data logged only tells one side of the story, since data is actually logged in order to be extracted later on to support operational decisions, which often have to be made within short time constraints; therefore data extraction must be as fast as possible.

It is perfectly legitimate for users to ask for data spanning long time periods and/or from long ago. Therefore the LS must satisfy such diverse requests not only as quickly as possible, but also whilst remaining stable such that is can support other operations in parallel.

The determining factor in how a service performs is *always* how the service is used. Experience has shown that there is often a big difference between how service providers *think* the service will be used, how users *claim* they will use the service, and how users *actually* use the service. This is where instrumentation comes in…

WHAT IS INSTRUMENTATION?

In this paper, instrumentation refers to capturing information about service activity in real time, and over time, in order to know who is doing what, from where, how things are being done, and how long various actions take.

Who?

This should always indicate the *real* end-user of the service – somebody who can be contacted. In other words, in an n-tier environment, it should not just be the directly connected OS user on one of the tiers.

What?

In its simplest form, this could be the name of a method / function / procedure etc. A more comprehensive solution would also capture details of all of the dimensions that can affect the outcome of an action and/or the performance of the service. These details are domain specific, but an example from the LS when querying data would be: the API method, the names of the signals concerned, the time window, and any additional data manipulation parameters (see Figure 4).

Where?

This should be a host name or IP address, and process id, which can be used to physically locate calls being made to the service.

How?

This means identifying which application (by name) is using the service, and if the service is accessed via libraries – which versions of the libraries are being used.

How Long?

Knowing the amount of time spent doing something is an essential ingredient in understanding how a service is performing, and why problems may have occurred. Therefore it is necessary to capture the elapsed time for each significant action executed within the service.

Besides these key elements, it is also important to instrument if actions finish successfully or throw exceptions in order to understand unexpected behaviour.

WHY INSTRUMENT?

Instrumentation is often considered an unnecessary overhead, especially by developers who want their code to run as fast as possible. However, this is a rather shortsighted view on things.

Knowing the answers to the questions above enables service providers to understand how a service is really being used (or misused), and how it is performing in terms of both throughput and response times. In turn, this allows to pre-empt problems, identify potential bottlenecks, plan system upgrades, and when issues inevitably occur – diagnose and react swiftly and effectively.

Collectively, these benefits far outweigh any perceived run-time overhead of having instrumentation in place.

The rest of this paper will focus on particular examples of instrumentation deployed in the LS, and how it has helped meet the requirements for performance, scalability, and stability.

DATA LOADING

Every day, the LS treats millions of data loading requests, coming from hundreds of client processes. The distribution of these requests across clients is heavily skewed. For example, one client may be responsible for

sending up to 40% of the data, and other just 1%. In order to know how the systems are being used, it is important to capture these data distributions.

Likewise, the data distribution within data loading requests may be heavily skewed across clients, or over time. In other words a fixed size data loading request may contain a lot of data for a few signals, or a small amount of data for many signals. This distribution can have a significant impact on performance, and therefore also needs to be captured to support performance analysis.

The other factor impacting performance of data loading in the LS is whether or not a request contains duplicate data (same timestamp received for the same signal), which requires a special treatment taking 4 times longer to process than a request without duplicate data.

Initial Implementation

The instrumentation in the LS has evolved significantly over many years. Initially most of the above details were just captured in log files. The problem with this approach is that they were very difficult to analyze, especially as the parallel load on the LS began to increase, and non-related log entries become more and more interleaved.

To understand the data distribution across clients, an internal database job ran queries against the logged data, making aggregates of the amount of data received per client. This approach was not scalable, and as the data rates increased the aggregate queries were continually adapted to use increasingly smaller sample periods of data. A new approach to instrumentation was required.

Evolution

A well-structured instrumentation framework was developed and put in place at the level of the data loading API running on the Oracle application servers. This framework captures all details of all data loading requests, performs on-the-fly in-memory data aggregations, and writes the results into the database on a daily basis.

This approach is extremely accurate (since aggregates are based on actual data rather than data samples), and avoids the need to use significant database resources to estimate system usage. In addition the time spent on each action (parse, check, prepare, load) within each data loading request is captured and aggregated to facilitate analysis of system performance, and identify bottlenecks and bad clients.

The other major advantage with the instrumentation framework is that all information is well structured and exposed via JMX using Java *managed beans* (MBeans), which can be consulted in real-time via any JMX (Java Management Extensions) interface. This allows service providers to easily see what the systems are currently doing, and diagnose on-going problems.

INTERNAL DATA TRANSFER

The majority of the data logged in the MDB are candidates to be transferred to the LDB for long-term storage. What data actually gets transferred is governed by a comprehensive set of metadata defining things such as deltas, smoothing, fixed logging, precision etc. for each of the defined signals. The act of applying the metadata to the raw data, and filtering and transferring the results to the LDB are carried out using in-house developed PL/SQL code which is executed in parallel by 8 internal database jobs running every 5 minutes. The signals whose data is treated by 1 of the 8 jobs are distributed across the jobs according to a predefined category for each of the signals.

Knowing how each execution of the data filtering and transfer jobs performs, in terms of number of signals, number of candidate values per data type, number of logged values per data type, and times taken for each internal action is essential.

Data Capture & Diagnostics

The PL/SQL data filtering and transfer code captures all of the above information in memory, and writes the results into dedicated database tables after each execution.

This detailed information remains available for 7 days (lifetime of MDB data) and is extremely useful for diagnosing performance problems – identifying if long executions times are isolated to particular groups of data, specific data types, certain times of the day or a specific type of action (such as data collection and filtering in the MDB, or data transfer to the LDB).

The detailed information is also aggregated on an hourly basis (Figure 3), and results are stored long-term in the LDB. This aggregate data helps identify trends in system performance such as correlations with accelerator performance, or gradual performance decreases as demands on the system increase (e.g. requests to log data for more signals and / or at higher frequencies.

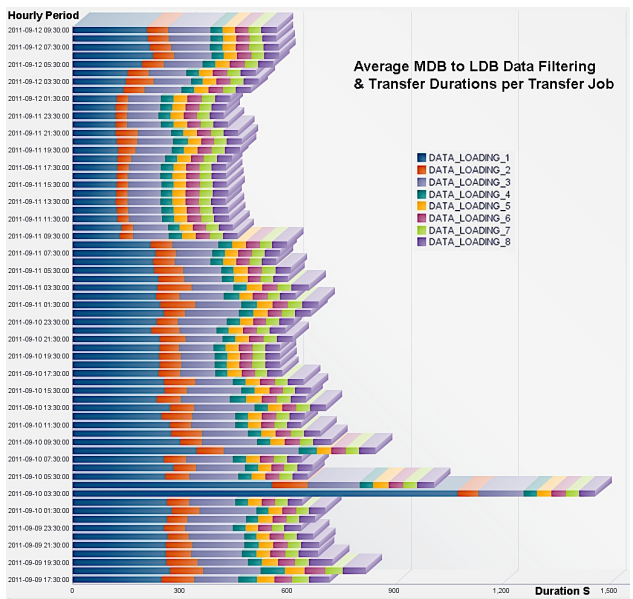


Figure 3: Example MDB to LDB instrumentation data.

DATA EXTRACTION

With up to 2 million requests per day to extract data for one or more signals over greatly varying time periods – data extraction from the LS represents a significant portion of overall activity.

Data rates vary significantly from one signal to another, and from one time period to another (e.g. according to whether or not there is beam present in the LHC).

In such an environment, users are often unaware of the amount of data that they are implicitly requesting, or of the best methods to use to extract with.

Aiming for Service Stability

As part of an attempt to assure service stability, every request to extract data is transparently instrumented, exposed, and logged using a framework similar to that for data loading, based on JMX. For each user: the running, last added, last finished, and last unsuccessful requests are always accessible via any JMX console. The *Who*, *What*, *Where*, *How*, and *How Long* information is embedded in each request, including signals involved, the extraction time window, invoked method, elapsed time, library versions, and the result (see Figure 4).

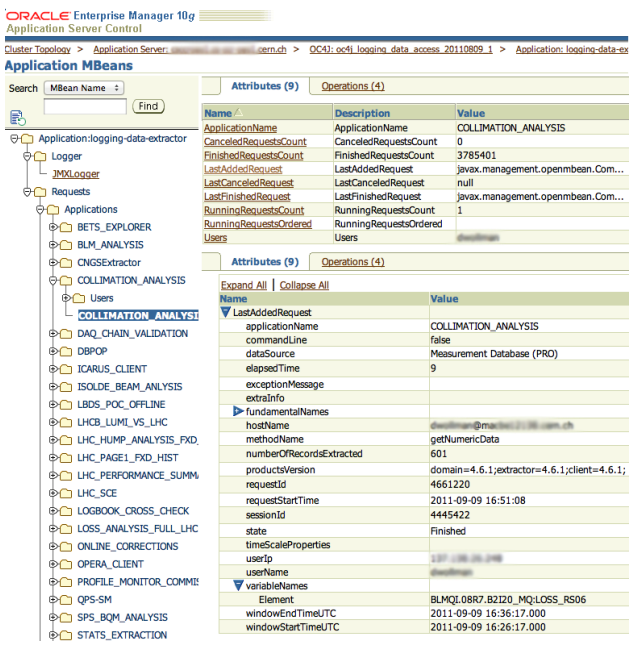


Figure 4: Data extraction instrumentation via JMX.

The ease of access of this information greatly facilitates following up support requests, since service providers can quickly access the full set of details of what the user is trying to do. Furthermore, because this information is accessible in real-time, a JMX agent connects every 5 seconds to the remote data extraction server, assesses the current situation, and can take various actions:

- If a request has been running for too long, a warning is first sent to service administrators, and if the situation continues – the request will be terminated. In such situations, it is common practice for the service administrators to diagnose the problem and pro-actively contact the user. More often than not – the users just need to be advised about which alternative methods to use or attribute values to apply.
- If any centralized data extraction server fails, service administrators are notified of the failure, together with details of all requests running prior to the failure, such that they can diagnose the cause, inform the user responsible, and adapt the service to be more resilient in the future.

Another way in which the captured data is used is related to backwards compatibility during upgrades to the API. Because all method calls are logged, it is possible to deduce whether or not certain users will be affected by necessary API changes, and contact them in order to adapt their code, or delay the changes.

SUMMARY

Instrumentation should not be considered as an overhead, but rather as an integral component of any software infrastructure. Once in place it quickly becomes part of the backbone of the system, allowing service providers to quickly and confidently diagnose problems, tune system performance, and plan upgrades.

The Logging Service instrumentation data is constantly used to support users, and has helped unravel otherwise impossible to diagnose problems in a complex and distributed environment.

The Logging Service is a stable, high performance, and heavily used service. The performance, proven ability to scale, and overall stability are testament to the value of the significant instrumentation in place.

REFERENCES

[1] C. Roderick and R. Billen, “Capturing, Storing and Using Time-Series Data for the World’s Largest Scientific Instrument”, November 2006, CERN-AB-Note-2006-046 (CO).

[2] C. Roderick et al., “The LHC Logging Service: Handling Terabytes of On-line Data”, ICALEPCS’09, Kobe, Japan, October 2009, WEP005.

CONTROL SYSTEM FOR CRYOGENIC THD/DT LAYERING AT THE NATIONAL IGNITION FACILITY*

M.A. Fedorov, O.D. Edwards, E.A. Mapoles, M.Y. Mauvais, T.G. Parham, R.J. Sanchez, J.D. Sater, B.A. Wilson, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550, USA

Abstract

The National Ignition Facility (NIF) is the world largest and the most energetic laser system for Inertial Confinement Fusion (ICF), located at the Lawrence Livermore National Laboratory (LLNL). In 2010, NIF began ignition experiments using cryogenically cooled targets containing layers of tritium-hydrogen-deuterium (THD) or deuterium-tritium (DT) fuel. The 68 μm-thick ice layer is formed inside of a 2 mm target capsule at temperatures of approximately 18.3 Kelvin. The ICF target designs demand sub-micron smoothness of the THD/DT ice layer. Precise formation and characterization of such layers is a challenging task and still an active research area. It requires a flexible control system capable of executing evolving layering protocols. At NIF, this task is performed by the Cryogenic Target Subsystem (CTS) of the Integrated Computer Control System (ICCS). ICCS is a large-scale, distributed control system which integrates scientific instruments, control hardware and computing platforms under a common object-oriented architecture. The CTS provides precise cryogenic temperature control, advanced x-ray imaging capability, and monitoring of vacuum and gases. Equipped both with automatic software engines and an interactive development environment, the recently deployed control system has enabled first NIF cryo-layered target campaigns and supported layering research.

INTRODUCTION

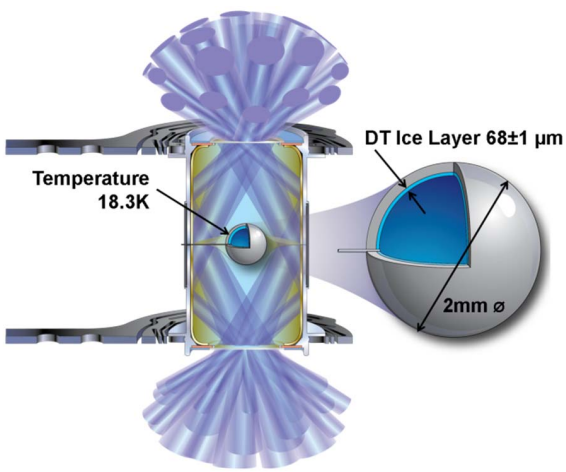


Figure 1: NIF cryogenic layered target.

The National Ignition Facility at Lawrence Livermore National Laboratory is a 192 beam, 1.8 MJ, 351 nm laser designed to conduct Inertial Confinement Fusion (ICF) experiments. Construction of NIF was completed in March of 2009. While NIF was conducting its first hohlraum energetics campaigns, the facility was also preparing for cryo-layered target experiments [1]. The key systems commissioned in the first half of 2010 included the Cryogenic Target Positioner (CryoTARPOS) with the Ignition Target Insertion Cryostat (ITIC) and the Load, Layer and Characterization Station (LLCS). Along with installation of hardware, the software for the Cryogenic Target Subsystem (CTS) of the Integrated Computer Control System (ICCS) was deployed and qualified for operations [2]. Precise cryogenic THD/DT layering was the main capability delivered with this new hardware and software.

An ignition target consists of four essential components: the hohlraum, the thermal control hardware, the capsule and the ice layer, see Fig. 1. The THD/DT ice layer is formed inside of the 2 mm target capsule at temperatures of approximately 18.3 K. In ignition experiments, the ice layer is 68 μm thick. Most of the target parts are precisely manufactured and calibrated in dedicated facilities, weeks and months before the shot. The ice layer is formed in the target mounted at the end of the target positioner, next to the NIF target chamber, just hours before the shot. Furthermore, the ice layer is cryogenic and is not accessible for direct inspection. However, the quality of the ice layer formation (or "manufacturing") is essential for the success of the ignition experiment.

THD/DT LAYER FORMATION

The THD/DT ice layer is formed by executing a cryogenic layering protocol. The layering protocols are first developed and proved in the R&D lab to accommodate a variety of target designs and fuel mixes. The protocols consist of several melting-freezing phases and last 17-24 hours. The x-ray images of the developing ice layer are periodically acquired in three imaging directions. The images provide the feedback on the layering process.

The goal of the layering protocol is to produce an ice layer which is both symmetrical and smooth. In a high-quality layer, the ice is distributed into a perfect sphere. The sphericity is measured by calculating the power spectrum of the unwrapped image in three directions. In addition, the ignition-quality layer must be free of local defects, such as boundary grooves and thermal stress cracks.

The formation of a spherical shape and elimination of local defects require application of different techniques.

* *This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.* LLNL-CONF-505491.

Beta-Layering

The general spherical shape is formed using the beta-layering process [3]. The beta-decay of tritium generates volumetric heat in the THD/DT ice. Thick regions of the ice layer have higher surface temperatures. The THD/DT fuel evaporates from the thick regions which are warmer and condenses in thin areas which are cooler. Given a symmetrical capsule, a uniform thermal environment and some time, the beta-layering will produce a spherical ice layer. The time constant of this process is about 27 minutes. Unfortunately, this technique is not efficient in preventing local defects.

Single Seed Crystal Growth

It was noticed that most of the local defects are due to the boundaries between ice segments grown from multiple seeds. Thus, a technique was developed which attempts to isolate a single seed and then grow the entire spherical layer from that seed [4,5]. The procedure involves several hours of precise temperature manipulations using x-ray image analysis as feedback. First, the capsule is rapidly cooled. The resulting ice layer formed inside of the capsule has an irregular shape and many defects. Next, the temperature is gradually increased to melt most of the ice inside the capsule. The melting process is monitored with the x-ray imaging system and is stopped when almost all of the fuel is liquid again. The remaining ice speck serves as a seed for the new layer formation. The temperature is decreased slowly, and the new layer grows as a single crystal over several hours.

CRYOGENIC TARGET CONTROL SYSTEM

Temperature Control

Both the beta-layering and the single seed crystal growth techniques require a highly uniform, and precisely controlled, thermal environment. At NIF, the Ignition Target Insertion Cryostat (ITIC) provides such a controlled environment [6], see Fig. 2. The ITIC is two meters long and it is mounted at the end of the NIF target positioner, as shown in Fig. 3. The target positioner is a large mechanical structure of approximately 15 meters long. The cryogenic operations start with the positioner fully retracted from the target chamber into an isolated vacuum vessel. The vessel is equipped with a three-axis x-ray target imaging system.

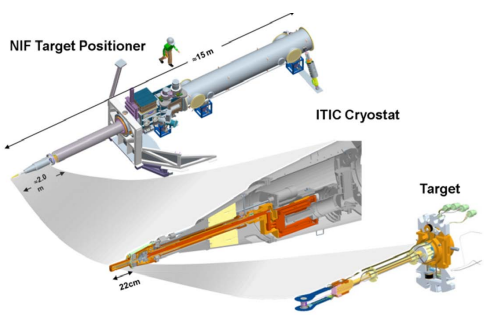


Figure 2: Ignition target cryostat system.

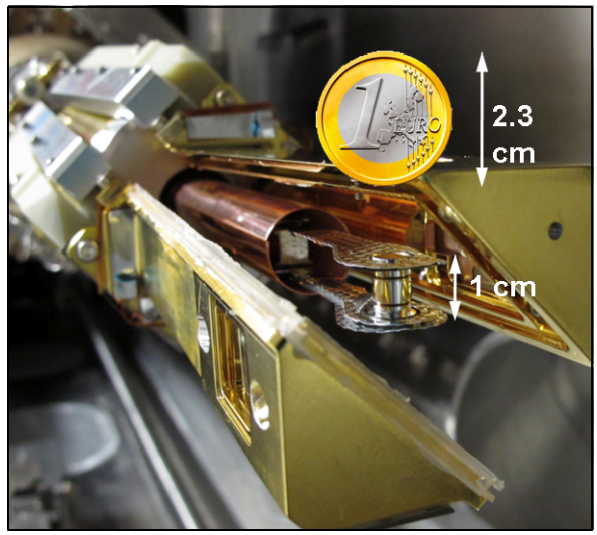


Figure 3: Front view of the ITIC shroud and the target.

Target temperatures are controlled down to 1 mK precision at multiple points. The top and bottom of the target have independent sets of controls. There are two additional shimming heaters in the middle section of the hohlraum. By adjusting these temperature control points, a highly uniform thermal environment can be created for the ice layer, even when the external environment or the target components are not perfectly symmetrical. The ice layer itself serves as a precise indicator of the temperature field. When the x-ray characterization of the layer reveals that it is asymmetric or not concentric, a calculated adjustment of the heaters is performed and beta-layering brings the layer into desired spherical shape.

The target base and the DT fuel reservoir are also precisely controlled. This is needed for filling the target with fuel. The reservoir and the target capsule are connected by a thin (10 μm OD, 5 μm ID) fill tube. The fuel is transferred by adjusting the temperature gradient between the target and the reservoir. Another important feature of the cryogenic system is the helium tank which provides thermal capacitance sufficient to maintain temperatures even when the mechanical cryogenic cooler is turned off for several minutes. This thermal capacitor capability is utilized to reduce vibration blur during the ice layer characterization.

Each experiment at NIF is conducted with a new target. There are several designs of cryogenic targets. They have different sets of control points. Each sensor comes with a unique calibration file. Therefore it is essential for the control system to be data-driven and support different control topologies.

X-ray Imaging System

The distribution of DT fuel inside of a target capsule is monitored and measured using a three-axis x-ray imaging system. The phase contrast, computer enhanced imaging reliably detects boundaries of solid and liquid DT inside of the beryllium and plastic capsules.

The imaging system includes three sets of x-ray sources and x-ray cameras arranged as three perpendicular axes.

The vertical axis is aligned with the hohlraum laser entrance hole and provides an unobstructed full view of the capsule. The two side views are taken through the starburst-shaped apertures in the hohlraum walls.

The image acquisition engine controls and monitors imaging hardware, see Fig. 4. It configures hardware settings, such as x-ray source voltage and beam current.

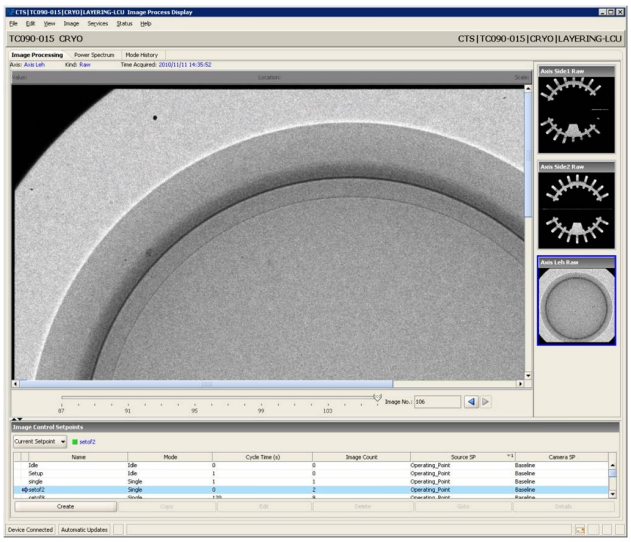


Figure 4: Control panel for x-ray imaging engine.

Because of the proximity of the target to the mechanical cryogenic cooler, the vibration blur distorts fine features of the images. To reduce blurring, the image acquisition engine times camera exposures to pauses in the cooler operations. The signal-to-noise ratio of the images is enhanced by taking multiple exposures and computing summed images. The capsule drift between the exposures is compensated by aligning images through image registration.

Interactive Layering Environment

In addition to automatic, data-driven software engines, the cryogenic controls are equipped with an interactive analysis and development environment. The environment is based on MATLAB and is implemented as a MATLAB Layering Toolbox extension. The environment enables real-time interaction with the growing ice layer, see Fig. 5.

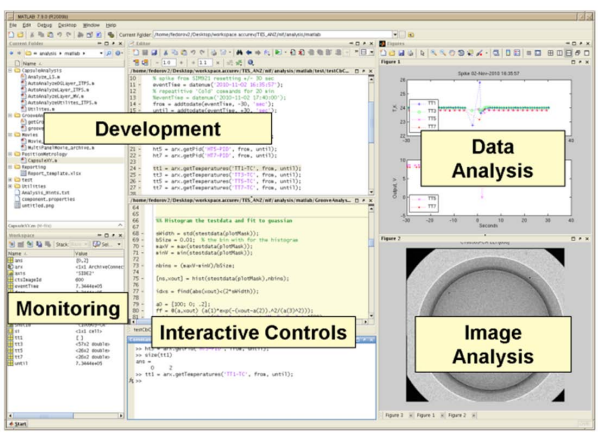


Figure 5: Interactive layering environment.

Implementation of the Layering Toolbox is based on the standard ICCS frameworks and CORBA middleware. Through this distributed control system infrastructure, the interactive environment automatically benefits in network connectivity, process concurrency and data sharing. The layering specialists at multiple workstations can monitor and support the ongoing layering process. Interactive sessions co-exist and communicate with the automatic software engines. The cryogenic process can be switched between automatic and interactive execution based on the complexity and novelty of a specific procedure.

Isolation of the interactive environment from the low-level hardware controls was one of the key design goals. The cryogenic target specialists preferred to avoid dealing with the specifics of the hardware configuration. The control system operators demanded that interactive environment actions are always entirely contained within a well-defined scope. The isolation goal was achieved by introducing the layering recipe controls abstraction. The minimalistic interface provides only a few virtual control points relevant to the physics of layering. It hides dozens of configuration settings which are managed by the core control system. The x-ray imaging is abstracted into named acquisition patterns and a flow of images. The low-level settings and sequencing of the x-ray imaging hardware are handled by the automatic software engines. The mapping of the virtual layering control points to hardware is data-driven and is managed by the core control system.

The development of the interactive layering environment was greatly simplified by choosing a well known scientific and technical computing product (MATLAB) as a COTS platform. While the Layering Toolbox codifies our domain expertise, the COTS component supplies a user-friendly IDE for code editing, debugging and version control. Extensive selection of data visualization and image processing tools for MATLAB is readily available, both commercial and open-source.

The rich toolset and flexibility of the layering environment have simplified evaluation of the new target designs and supported layering research.

COORDINATION WITH NIF LASER SHOT

Layering Report

When the ice layer is completely formed, an extended set of image analysis routines is run to produce the Layering Report., shown in Fig. 6. The Report contains metrics and checks whether the layer meets the ignition specification. The Layering Report is a key decision point, when the scientific review board meets to decide whether to proceed with a NIF shot.

For visual inspection, the layering report contains unwrapped images of the ice layer in three dimensions. These images help evaluate the general spherical shape and severity of local defects. The power spectral density of the unwrapped ice surface is calculated for three axes

and it is plotted against the ignition target specification. In the Layering Report section shown on Fig. 6, the spherical shape is within the specification for all three directions. For local defects, a cumulative metric is calculated based on the total surface area affected by the defects.

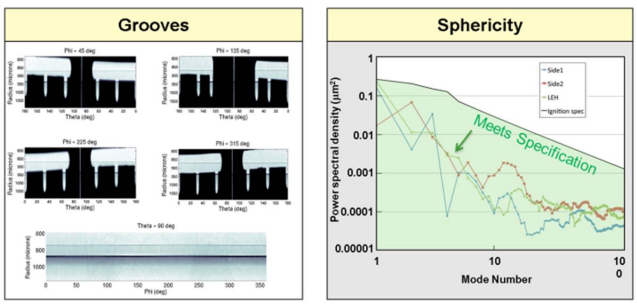


Figure 6: Section of a Layering Report.

The summary of the Layering Report includes a recommendation to proceed with the shot or to re-attempt the layering.

Cryogenic Activity and NIF Shot Sequence

Formation of the high-quality ice layers is a long process. It takes many hours, see Fig. 7. Including retries and supporting operations, it can take days. All cryogenic operations are coordinated under the Cryogenic Activity. The Cryogenic Activity is a sequence of steps similar to a NIF Shot [7]. Due to its extended duration, the Cryogenic Activity starts independently from a NIF shot. When the layer is close to completion, the NIF shot sequence starts. The Cryogenic Activity communicates to shot controls and joins the NIF Shot cycle.

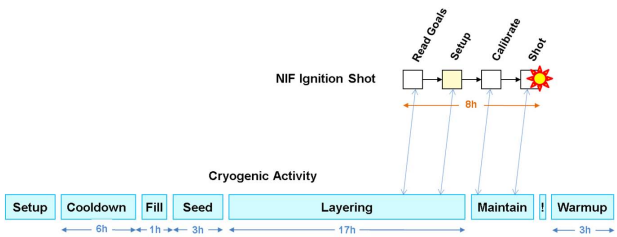


Figure 7: Timeline of Cryogenic Activity and NIF Shot.

Final Pre-shot Cryogenic Target Operations

The cryogenic target subsystem continues to monitor target temperatures during target insertion into the chamber and during target alignment. Seconds prior to the main laser shot the cryogenic controls perform final temperature adjustments and open the shroud.

Ignition target specifications require that the final target temperature is ~1.5K lower than the temperature at which the ice layer is formed and maintained [4]. At lower temperatures, the ice layer quickly develops roughness and cracks. To minimize the degradation of the ice surface, the cryogenic controls perform final temperature adjustment, called quench, only 30 seconds prior to the main laser shot (T-30). To further reduce the thermal stress, the quench is performed as a smooth linear ramp.

From the beginning of the cryogenic activity until the laser shot, the protective shroud surrounds the cryogenic target. This shroud shields the target from infrared radiation and prevents undesirable condensation on the outer surfaces of the target. The NIF shot control software commands mechanical opening of the shroud eight seconds prior to the main laser shot (T-8). The cryogenic system has adequate control authority to maintain precise temperatures for several minutes after the shroud is open. This is sufficient to handle normal and off-normal shot cases.

Normally, at T-0 the main laser shot is fired and the cryogenic target is consumed. Next, the cryogenic controls proceed to the warm-up sequence. If the laser shot is aborted during the final seconds, the cryogenic controls quickly revert the quench and return the target to temperatures at which the ice layer can be maintained safely until the next attempt.

RESULTS

Since 2010, the Cryogenic Target System supported 72 NIF experiments, including 13 with layered targets. The integrated hardware-software system provided a robust and accurate platform for cryogenic target experiments. The automatic data-driven process consistently executed complex layering protocols. Finally, the flexible interactive environment simplified evaluation of new target designs and supported layering research.

REFERENCES

[1] J.D. Lindl et al., “Progress towards ignition on the National Ignition Facility,” Nucl. Fusion 51, 094024 (2011).

[2] J. Fisher, “Status of the Integrated Computer Control System,” IAEA 8th Technical Meeting, San Francisco, http://iaea-tm2011.org (2011).

[3] J.K.Hoffer and L.R.Foreman, “Radioactively induced sublimation in solid tritium,” Phys. Rev. Lett. 60, 1310-1313 (1988).

[4] S.O.Kucheyev and A.V.Hamza, “Condensed Hydrogen for Thermonuclear Fusion,” J. Appl. Phys. 108, 091101 (2010).

[5] Chernov, A. A. Kozioziemski, B. J. Koch, J. A. Atherton, L. J. Johnson, M. A. Hamza, A. V. Kucheyev, S. O. Lugten, J. B. Mapoles, E. A. Moody, J. D. Salmonson, J. D. Sater, J. D., “Single crystal growth and formation of defects in deuterium-tritium layers for inertial confinement nuclear fusion,” Appl. Phys. Lett. 94, 064105 (2009).

[6] Gibson C.R.; Atkinson D.P.; Baltz J.A.; et al., “Design of the NIF cryogenic target system,” Fusion Science and Technology v. 55, no. 3, pp 233-236, 2009

[7] D. Mathisen, “Orchestrating Shots for the National Ignition Facility,” IAEA 8th Technical Meeting, San Francisco, http://iaea-tm2011.org (2011).

CONTROL AND TEST SOFTWARE FOR IRAM WIDEX CORRELATOR

S. Blanchet, D. Broguiere, P. Chavatte, F. Morel, A. Perrigouard, M. Torres
IRAM, Grenoble, France

Abstract

IRAM[1] is an international research institute for radio astronomy. It has designed a new correlator called WideX for the Plateau de Bure interferometer (an array of six 15-meter telescopes) in the French Alps. The device started its official service in February 2010. This correlator must be driven in real-time at 32 Hz for sending parameters and for data acquisition. With 3.67 million channels, distributed over 1 792 dedicated chips, that produce a 1.87 Gbits/sec data output rate, the data acquisition and processing and also the automatic hardware-failure detection are big challenges for the software. This article presents the software that has been developed to drive and test the correlator. In particular it presents an innovative usage of a high-speed optical link, initially developed for the CERN ALICE experiment, associated with real-time Linux (RTAI) to achieve our goals.

INTRODUCTION

After the installation of its new generation of receivers in 2006, IRAM has started to build a new correlator called WideX that can process up to 64 GHz of total analog bandwith [1]. It is a big machine with 3.67 million channels, distributed over 1 792 custom[2] chips. With such a number of components, powerful programs for automatic testing are crucial, otherwise the project fails. Real-time driving at 32 Hz and data processing is also a serious software challenge because the total device output data rate is about 1.87 Gbit/s.

TECHNICAL CHOICES

Data Transmission

For technical convenience the correlator is divided into four identical sub-units (Fig. 1), driven by four computers. It means that it requires 4 data links at 448 Mbit/s. Finding a way to transfer data at such rate, was a headache for engineers. Fortunately CERN had already developed such a high-speed serial link for its own needs[3] and it was possible to buy this technology jewel at a very affordable price. The CERN solution is called Detector Data Link (DDL) [2]. It is a versatile high-speed optical link, that can transfer up to 1.5 Gbit/s. It is composed of:

[1]Institut de Radioastronomie Millimétrique

[2]The correlator chip is a 250 MHz application-specific integrated circuit (ASIC) designed under IRAM specifications [1].

[3]CERN has developped DDL for the LHC/ALICE experiment.

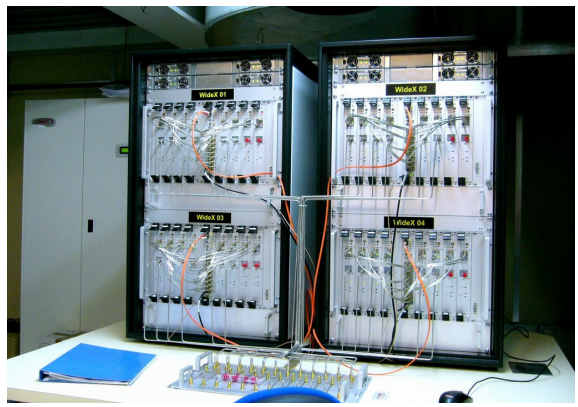


Figure 1: IRAM WideX on the observatory site. Note the four orange fibers that connect the four sub-units.

- a *Source Interface Unit* (SIU) to plug into the correlator (Fig. 2).

Figure 2: CERN Source Interface Unit board.

- a *Destination Interface Unit* (DIU) hosted on a readout receiver card (Fig. 3) to connect to a computer.

Figure 3: CERN Read-Out Receiver Card (PCI-X slot) with 2 Destination Interface Units.

- a fiber optic to link directly the SIU and the DIU (orange fibers on Fig. 1).
- a Linux driver with the source code.

This optical link is very easy to use: the data are sent to the SIU through a parallel bus, then they appear automati-

cally in the computer memory at the address that has been provided to the driver.

Real-time Operating System

While processing large amounts of data, the computer must drive the data acquisition at 32 Hz, therefore a real-time operating system is required. Linux-RTAI[4] was chosen because it has already been successfully used at IRAM, for critical tasks like the antenna control.

This real-time engine guarantees that the data acquisition runs at 32Hz, without losing any data, whatever the computer load is. Among the different RTAI running modes, LXRT[5] was prefered, because it allows hard real-time programs to run in user space: the software development is easier and programs can be written in C++ instead of C only.

The only small drawback is that the CERN driver and libraries are not directly compatible with RTAI, but a modification of the CERN source code has solved this problem.

Software Libraries

Because electronic engineers need very fast and responsive testing tools, the fastest software libraries have been carefully selected to build the programs:

- Qt: *a free C++ cross-platform framework* [4]. It provides fast and nice graphical widgets and also many other very useful classes. It is developped by Nokia Corporation.
- Qwt: *Qt Widgets for Technical Application* [5]. It is a Qt extension, that provides nice technical widgets like a plotter (QwtPlot) with built-in zoom and autoscaling. It is a community project.
- FFTW: *Fastest Fourier Transform in the West* [6]. It is a free library to compute the Discrete Fourier Transform in one or more dimensions. It uses code-generation and runtime self-optimization techniques to achieve a very high level of performance. This software library was developed at the Massachusetts Institute of Technology (MIT).

SOFTWARE

The different pieces of hardware impose some constraints on the time diagram (Fig. 4) that should be explained to understand the most interesting details of the software design.

1. The correlator data output is scheduled on an internal 32 Hz clock[6].
2. To read data, the computer sends a `RDYRX` (Ready-to-Receive) command before the 32 Hz pulse. The transfer starts with the 32 Hz pulse.
3. The data transmission is long and uninterruptible. Therefore it allows only a small sending window after `EOBTR` (End Of Block Transfer).
4. The CERN board generates no hardware interrupts: a polling is required to detect the end of transmission (`EOBTR`).
5. The CERN board writes data directly in memory without using the central processor, therefore it is the best moment to process data from the previous acquisition period.

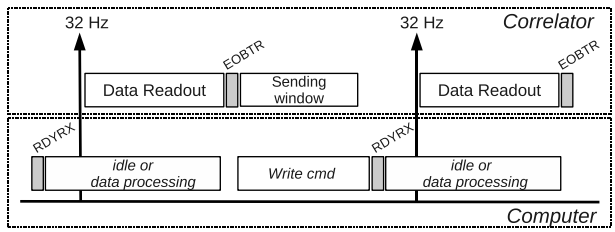


Figure 4: Time diagram.

Time Synchronization

Like the antenna position, the correlator acquisition parameters change relentlessly with the Earth rotation. Unlike the correlator that synchronizes itself directly on the time signal of the observatory maser, the computer has no particular hardware for clock synchronization. However a software-only solution works very well:

1. Because the transmission duration is constant and starts always on a 32 Hz pulse (Fig. 4), the computer considers the end-of-block-transfer (`EOBTR`) event as a faithful image of the correlator clock. Therefore the real-time tasks are synchronized on this event. The polling loop starts only when required, at the very last time, and exits after only few iterations. In this case, it makes sense to use a high-frequency[7] loop.
2. To know the absolute time associated to each `EOBTR` event, the computer uses the Network Time Protocol[8] server that depends on the observatory maser.

Software Architecture

There is only one real-time program: `CONTROL` (Fig. 5). It has two real-time threads: `DRIVE` and `PROCESS`.

[4]RTAI: RealTime Application Interface [3]. It is a real-time extension for the Linux kernel from the *Department of Aerospace Engineering, Politecnico di Milano, Italy*.

[5]LXRT: Linux Extension for Real Time. It is a special RTAI scheduler.

[6]The 32 Hz frequency is dictated by the Walsh functions that are used to reduce the effects of electrical crosstalk between antenna signals.

[7]In our case, the polling loop runs at 1 kHz, but it could be higher.

[8]Network Time Protocol (NTP) is a protocol for synchronizing the clocks of computers over variable-latency networks.

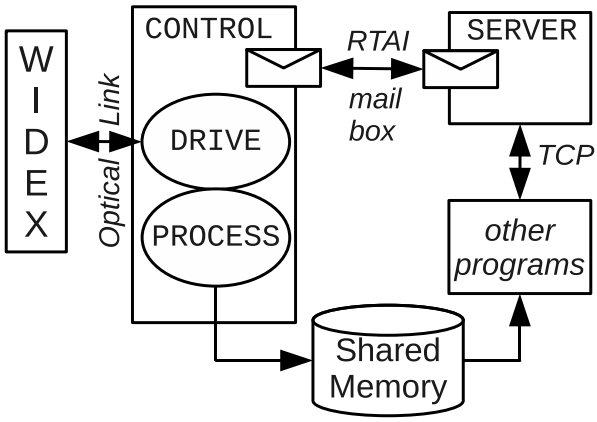


Figure 5: Software architecture.

- `DRIVE` drives the data acquisition. It has the first real-time priority to avoid any data loss. It executes the following loop of instructions:
  1. Get a free memory page[9], and send its address to the CERN driver.
  2. Send a `RDYRX` command to the correlator, and then sleep[10] until just before the next `EOBTR` event.
  3. Poll until `EOBTR` occurs.
  4. Push the page of the just-arrived data into the waiting queue of the `PROCESS` thread,
  5. Send any pending parameters to the correlator.
  6. Sleep until just before the next 32 Hz pulse.
  7. Repeat the loop.
- `PROCESS` processes the raw data from the correlator. It has the second real-time priority to guarantee that the processing is finished in the expected time. It executes the following loop of instructions:
  1. Take the first data page from its waiting queue.
  2. Process the data and write the result into the shared memory.
  3. Push back the memory page to the pool.
  4. Repeat the loop.

Because of the chosen priorities, the Linux kernel and all the other software run only when `DRIVE` and `PROCESS` are asleep. To send specific command to the correlator, programs must pass through `SERVER` (Fig. 5) to transform TCP requests into RTAI messages. The command will be really sent to the correlator during the next sending window (Fig. 4).

[9]The memory page comes from *physmem*: it is a reserved memory area for CERN DDL direct memory transfers. A realtime-compatible memory allocator has been specifically written for this area because the Linux kernel does not manage it.

[10]The sleeping and the transmission last exactly the same time, but the sleeping ends before, because it starts before.

RUNNING MODES

There are different running modes, one for each testing stage.

Simulation Mode

The target of this mode is to test software without needing hardware, and also to prove that the processing algorithms are correct. The simulation mode changes the behaviour of the driving task: instead of reading the CERN driver output, the thread calls real-time functions that generate custom simulation data.

There are several simulation patterns, one per kind of tests. For example, Fig. 6 shows a simulation pattern for chip failure detection.

Chip Debugging Mode

The goal of this mode is to check that all the correlator chips work as expected. This mode modifies the processing task: instead of analyzing spectrum, it compares together the raw outputs of the chips. A special program displays the results as a 448-cells matrix (Fig. 6). When a problem is detected in a chip, the cell color changes. The user can click on the chip to display its channel values (Fig. 7).

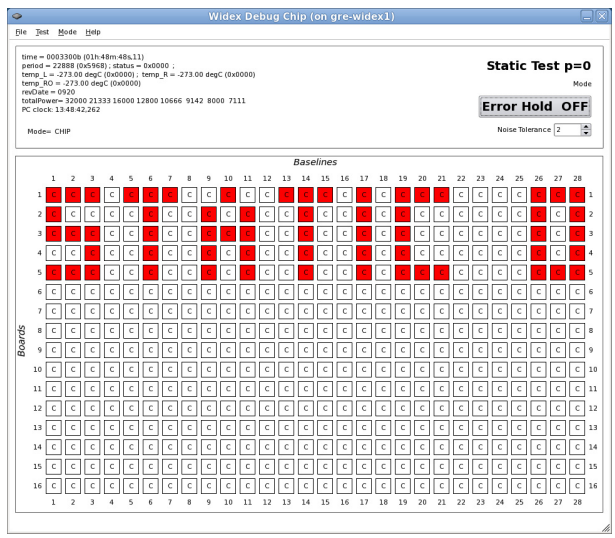


Figure 6: Chip debugging window. A simulation pattern is printing messages on the chip matrix.

Observing Mode

The purpose of this mode is to show that the correlator can extract the hidden signal from the noise. Therefore it implements almost all the operations for real observing. This mode modifies the processing task to analyze the spectrum in real-time. These signal processing operations are organized in sequence: Fast Fourier Transform followed by several corrections of levels and phases. Each pipeline stage can be enabled or disabled in order to measure its

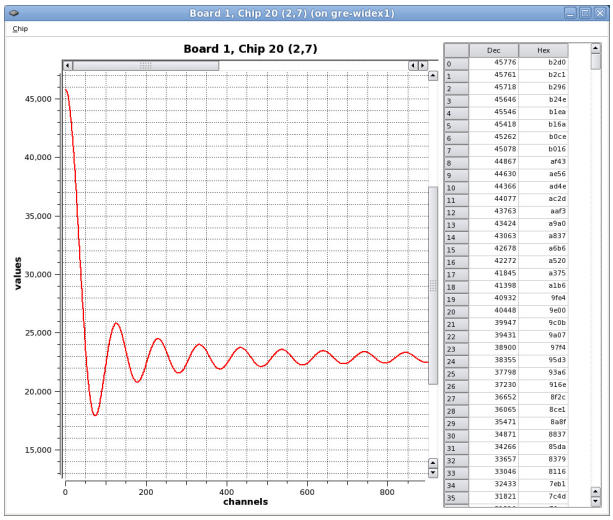


Figure 7: Content of a chip. The curve is plotted with QwtPlot [5].

individual effect and its contribution to the total execution time. A fast viewer (Fig. 8), with direct buttons to control signal processing, has been written to display the results. It heavily uses Qt [4], Qwt [5], and FFTW [6].

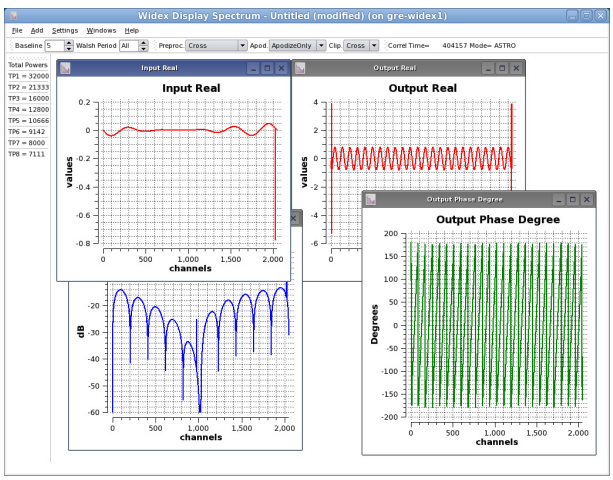


Figure 8: Dedicated viewer for correlator spectrum.

CONCLUSION

Thank to the innovative pair CERN DDL – RTAI, it is possible to build a very high-performance control software. It drives in real-time a complex scientific device, while processing large amounts of data. The icing on the cake is that the solution is built with free open-source software only.

ACKNOWLEDGMENT

The authors would like to thank Ervin Dénes, Csaba Soós and Pierre Vande Vyvre from CERN: we are grateful for their free hardware loan and their precious help at the early stage of the project.

REFERENCES

[1] M. Torres, “Main Technical features of the WideX correlator” http://www.iram.fr/widex

[2] CERN DDL website http://cern.ch/ddl

[3] RTAI website http://www.rtai.org

[4] Qt website http://qt.nokia.com

[5] Qwt website http://qwt.sourceforge.net

[6] FFTW website http://www.fftw.org

A NEW FAST DATA LOGGER AND VIEWER AT DIAMOND: THE FA ARCHIVER

M.G. Abbott, G. Rehm, I.S. Uzun, Diamond Light Source, Oxfordshire, UK

Abstract

At Diamond Light Source, position data from 174 Electron Beam Position Monitors (BPMs) and a number of X-Ray BPMs is distributed over the Fast Acquisition communications network at an update rate of 10 kHz; the total aggregate data rate is around 15 MB/s. The data logger described here (the FA Archiver) captures this entire data stream to disk in real time, re-broadcasts selected subsets of the live stream on demand to interested clients, and allows rapid access to any part of the saved data. The archive is saved into a rolling buffer allowing retrieval of detailed beam position data from any time in the last four days. A simple socket-based interface to the FA Archiver allows easy access to both stored and live data from a variety of clients. Clients include a graphical viewer for visualising the motion or spectrum of a single BPM in real time, a command line tool for retrieving any part of the stored data by time of day, and Matlab scripts for exploring the dataset, helped by the storage of decimated minimum, maximum, mean, and standard deviation data.

INTRODUCTION

Diamond uses Electron Beam Position Monitors for electron beam diagnostics and orbit position feedback. Each BPM button block is connected to a Libera [1] beam position processor, comprising RF processing for each of four pickup buttons, with ADCs and an FPGA which reduces the data to streams at a variety of data rates, and an embedded processor providing an EPICS interface [2].

In the Libera FPGA the sampled button intensities are converted to X, Y positions and reduced by decimation and filtering to machine revolution frequency (534 kHz) and then to the "Fast Acquisition" (FA) data rate of 10 kHz and the "Slow Acquisition" rate of 10 Hz for EPICS clients. The FA data is broadcast through the Diamond Communication Controller (CC) network [3, 4] which makes this data stream simultaneously available to all connected machines. The CC network allows machines to be freely added as receivers or transmitters with little or no impact on the operation of the network; the FA Archiver is one such machine.

The FA archiver receives CC updates at the FA data rate of 10 kHz, each update consisting of position information for all connected BPMs together with data from a handful of other data sources. This data is stored to disk in a rolling multi-day buffer and made available to client applications throughout Diamond.

The archiver runs on a dedicated server and writes to local disks; at Diamond we use a 6 Terabyte archive (two dedicated 3 TB disks as a 6 TB RAID 0 volume) giving us just under 4½ days of archive of 182 CC ids. The archiver acts as a socket server supporting a simple command language; commands are provided for subscribing to any subset of the live data stream, and for retrieving any part of the archived dataset.

Live data is available from the FA archiver both at the full 10 kHz data rate and decimated by filtering down to 1 kHz. Archived data is available at the full data rate or decimated binned by factors of 128 or 16384 (by default) to provide an overview of the stored data.

FA ARCHIVER AND CC NETWORK

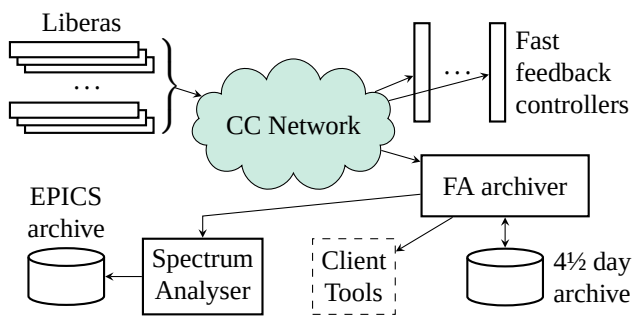


Figure 1: FA Archiver and CC network in context.

The Communication Controller network was originally constructed to make the 10 kHz FA data from each Libera BPM available for the operation of the fast feedback controllers which maintain position stability of the electron beam. The network synchronously distributes beam position X and Y coordinates every 100 µs with a propagation delay of around 50 µs.

The design of the CC network with a highly redundant topology broadcasting by storing and forwarding makes it easy to add new nodes to the network, both as contributors and as passive listeners. Other sources contributing to the network are a handful of X-ray BPMs and power pick-ups from the RF cavities. The FA archiver acts as a passive listener and makes the FA data freely available, see Fig. 1.

The archiver stores data at the full data rate, referred to here as the "FA" rate, and at two decimated rates, "D" data decimated by a factor of 128, and "DD" "double decimated" data decimated by a factor of 16834.

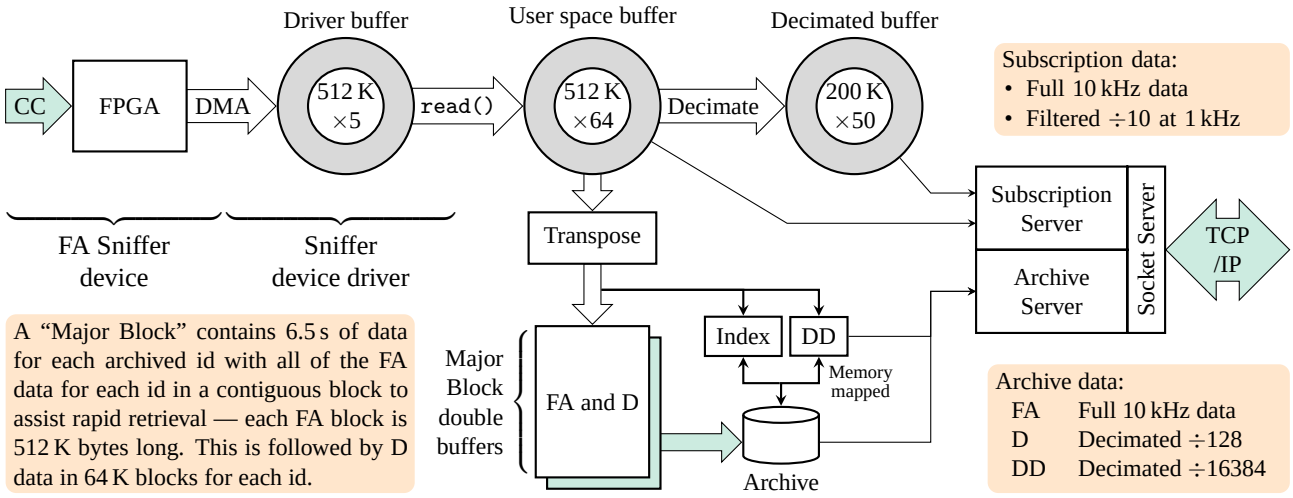


Figure 2: Architecture of FA archiver showing device, driver and archiver process writing to disk and serving clients.

ARCHIVER ARCHITECTURE

The FA archiver consists of the following components:

FA Sniffer. This is an implementation of the Communication Controller on a Virtex-5 FPGA PCI Express (PCIe) board. The sniffer captures communication frames and transfers them by DMA to memory.

Sniffer device driver. The device driver communicates with the sniffer card and makes the data stream available as a Linux device node, `/dev/fa_sniffer0`.

Archiver process. The archiver process reads frames from the device driver, saves them to disk after processing to optimise subsequent reading, and provides a socket server for client access to both the live data stream and a ÷10 decimated stream, see Fig. 2.

Client tools. A variety of client tools are able to connect directly to the archiver server, including a data visualisation tool (`fa-viewer`) for viewing the live data, a general purpose command line tool (`fa-capture`) for reading from the archive, and a Matlab script (`fa_zoomer`) for exploring the archive.

FA Sniffer

The FA Sniffer is so called because it passively "sniffs" FA data from the communication controller network. The FPGA design integrates the Diamond Communication Controller [3] into a bus master PCI Express architecture using the commercially available Xilinx Virtex-5 FPGA ML555 development platform [5]. The design, see Fig. 3, consists of target logic, DMA initiator logic, status/control registers and the Virtex-5 endpoint core for PCI Express. Target logic is responsible for capturing single double-word memory write and memory read PCIe Transaction Layer Packets (TLPs) for control and status register access. The DMA initiator logic generates memory write TLPs to transfer 2 K byte frame data from the communication

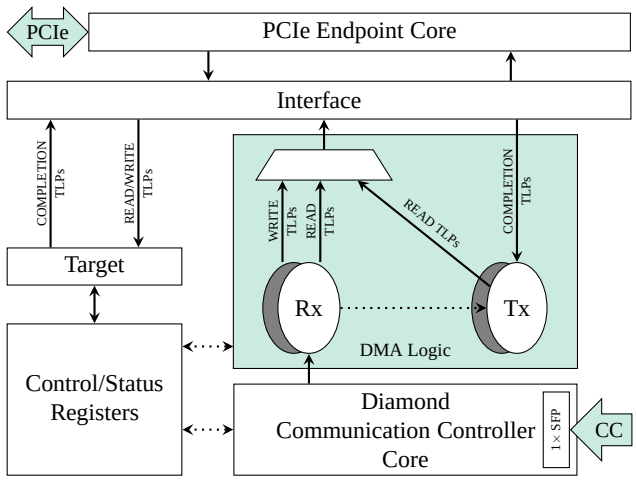


Figure 3: Architecture of the FA sniffer FPGA.

controller core to the host's system memory. The complete design occupies 30 % of bit slices and 20 % of block RAMs available on the FPGA.

The communication controller core captures a complete frame from the CC network on every communication controller cycle, with a new cycle starting every 100 µs. The frame transferred to memory by the sniffer is 2048 bytes consisting of 256 samples, each a pair of four-byte numbers corresponding to measured X and Y positions for FA ids 1–255 together with a 32-bit timestamp counter repeated in position 0 as "id 0".

The sniffer card maintains a queue of up to two DMA targets, address and frame count, one being written to, the next to be loaded when the first target is filled, at which point an interrupt is raised to the host machine. At this point the device driver is responsible for setting up a fresh buffered DMA target, otherwise the sniffer will halt.

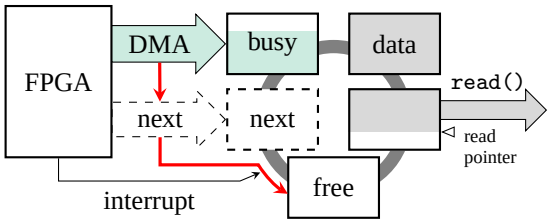


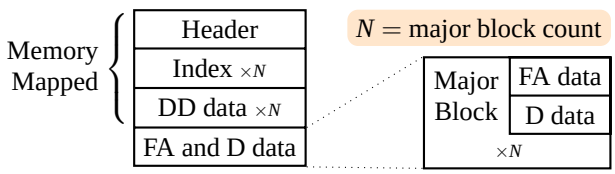
Figure 4: Architecture of the sniffer device driver.

Sniffer Device Driver

The sniffer device driver, see Fig. 4, maintains a circular pool of buffers for the FA sniffer to write to, by default five buffers of 512 K bytes each, enough storage for 40 ms of data per buffer. When the sniffer is running, two buffers are allocated to the sniffer and the remaining buffers contain data for the user space program to read or are free for the next DMA transfer. On each interrupt the pool cycles round by one buffer unless there is no free buffer, in which case the sniffer is allowed to halt.

Data capture is also interrupted during CC network synchronisation or if a communcation error occurs. Any interruption is signalled by the driver as "end of file" and recorded by the archiver as a gap in the archive.

The Archive on Disk



Per major block, repeated for each archived FA id:

FA data:	x, y	$\times 65536$
D/DD data:	$\overline{x}, \overline{y}, \lfloor x \rfloor, \lfloor y \rfloor, \lceil x \rceil, \lceil y \rceil, \sigma_x, \sigma_y$	$\times 512/\times 4$

Figure 5: Layout of archive store on disk.

The FA data stream from the sniffer is uniform, with constant sized updates received at an unchanging interval. The only variation in this structure is the occasional presence of gaps in the data stream where synchronisation of the communication network has been performed.

This uniform data format allows the archive to be stored as a simple fixed-format file, see Fig. 5. For performance it is better for the archive to be stored directly on the underlying block device, as filesystem block management otherwise adds a significant overhead for very large files.

Early experiments with the archive revealed that the optimum block size for transfers to and from disk is around 512 K bytes and so the layout of the archive on disk is optimised for reading by placing the data for a single FA identifier into blocks of 512 K bytes or 65536 samples each, or around 6½ seconds of data.

At the same time, to help with generating an overview of the entire archive, data is binned into 128 and 16384 sample bins in which the mean, minimum, maximum, and standard deviation values are computed. The decimated $\div 128$ data is stored together with the FA data as a "major block", the $\div 16384$ data is stored separately in a memory mapped area.

These major blocks are indexed by timestamps recorded in the index which is searched by binary search when an archive read request is processed.

A fixed size (4096 byte) header defines all of the operating parameters of the archiver (block sizes, decimation factors, etc.) together with the list of FA ids archived, and also records the current active block.

Data Processing

Two major blocks are maintained in memory, one being prepared as frames are received from the FA sniffer, the other being written to disk. FA sniffer frames are transposed (in blocks of 256 frames) into the major block layout.

To ensure that reads from the archive do not delay writes there is a limit on the number of reads that can be requested simultaneously, and there is also an interlock to prevent a read being initiated while writing to disk is in progress.

Filtered Decimation

For some applications of the archiver live FA data feed at the full 10 kHz data rate is too much; for example most of the environmental disturbances on the beam have their main effects in the bottom few hundred Hertz. In particular, the full orbit Spectrum Analyser (described below) only covers frequencies up to 320 Hz.

Thus the archiver decimates the entire FA data stream by a factor of 10 using a compensated CIC filter [6, 7], and this data stream is available as an alternative data source for subscribers. The spectrum of the decimated data is flat to ± 0.25 dB with better than 100 dB alias rejection up to 350 Hz.

APPLICATIONS

Archive Retrieval. A command line tool `fa-capture` provides full access from the command line to the dataset stored on the FA archiver, allowing any part of the archive to be retrieved and saved in either raw or Matlab format. Start and end times or start and sample count can be specified in a variety of formats.

Archive Exploration in Matlab. A Matlab tool called the `fa_zoomer` allows the archive to be explored interactively. An interactive window displays the current selection and allows any part of the selection to be zoomed and retrieved in more detail from the archive. The initial window uses "double decimated" data to show an overview of either the last 24 hours or the entire available archive (4½ days), and the decimation is reduced as zooming is refined.

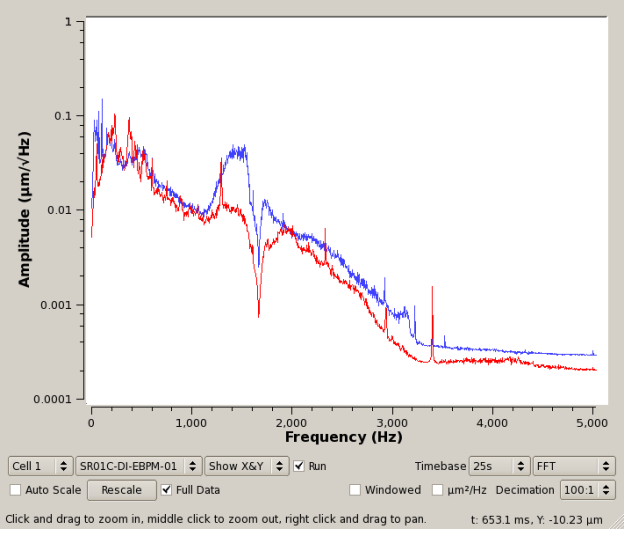


Figure 6: FA viewer application showing beam motion power spectrum of selected BPM over the last 25 seconds.

Live Visualisation. Figure 6 shows the `fa-viewer` tool, a Python Qt application which visualises the motion of the beam at any location. The user can choose to observe any available FA identifier which is then displayed as either a live history display (for up to 60 seconds of history) or a variety of spectral displays: FFT on log y or log xy scales, or an integrated power spectrum. Graphing is done using PyQwt [8] which is both fast and flexible, making it straightforward to develop a large variety of visualisations.

Audio Playback. As the beam disturbance frequencies as processed by the FA archiver are comfortably audible it was natural to try playing back the beam position as sounds. This turns out to work very well, and is occasionally quite instructive. Both topup and injection have very distinctive sounds, but what is more interesting is the variety of very odd sounds that appear to be superimposed on the beam by disturbances in the RF system.

Long Term Spectral Analysis. A dedicated server subscribes to the 1 kHz complete live orbit data and computes averaged power spectra in 1 Hz bins averaged every five minutes covering the spectral range from 1 to 320 Hz. Complete waveforms for all frequencies and all 174 BPMs are generated as EPICS PVs, both as power density spectra for each BPM and orbit disturbance for each frequency, and a limited selection of these waveforms is archived in the long term EPICS archive.

This allows a detailed examination of changes in the spectral behaviour of the machine over extended periods; it is easy to look at the spectrum for a week's operation of the machine and identify long term changes in disturbance.

Other Users. The sources for the archiver and driver can be downloaded from the Diamond web site [9], and the FA archiver has been used at Soleil since May 2011.

CONCLUSIONS

The motivation for the FA Archiver has been to gain access to longer periods of synchronously recorded position data from all channels (the whole orbit) at FA data rate. Previously our setup had only been able to capture ten seconds of this data using buffers inside the fast feedback processors. Having access to records of up to several days of this data facilitates the following:

- Identification and detailed investigation of any orbit events that happened in the recent past. To this end, the maximum/minimum/standard deviation calculations on the ÷16384 decimated data are invaluable.
- Correlation of orbit events with other measurements by adding X-ray BPMs and RF cavity voltage readings to the FA network [10].
- Modal analysis of orbit motion over longer periods. This is useful for the optimisation of the fast orbit feedback and for the identification of BPMs or correctors with subtle malfunctions.

Together with the long term spectral analysis all these methods allow monitoring and improvement of the reliability and performance of the fast orbit feedback system and ultimately the beam stability.

REFERENCES

[1] Instrumentation Technologies, `http://www.i-tech.si`.

[2] M.G. Abbott, G. Rehm, I.S. Uzun, "The Diamond Light Source Control System Interface to the Libera Electron Beam Position Monitors", ICALEPCS 2009.

[3] I.S. Uzun, R. Bartolini, G. Rehm, J.A. Dobbing, M.T Heron, J. Rowland, "Initial Design of the Fast Orbit Feedback System for Diamond Light Source", ICALEPCS 2005

[4] M.G. Abbott, J.A. Dobbing, M.T. Heron, G. Rehm, J. Rowland, I.S. Uzun, S. Duncan, "Diamond Light Source Electron Beam Position Feedback: Design, Realization and Performance", ICALEPCS 2009.

[5] Virtex-5 LXT ML555 FPGA Development Kit for PCI Express, `http://www.xilinx.com/products/boards-and-kits/HW-V5-ML555-G.htm`

[6] E.B. Hogenauer, "An economical class of digital filters for decimation and interpolation", IEEE Transactions on Acoustics, Speech and Signal Processing, pp. 155–162, April 1981

[7] Altera Corporation, "Understanding CIC Compensation Filters", AN-455-1.0, `www.altera.com/literature/an/an455.pdf`

[8] `http://pyqwt.sourceforge.net/`

[9] `http://controls.diamond.ac.uk/downloads/other/fa-archiver`

[10] G. Rehm, M.G. Abbott, C. Bloomer, I. Uzun, "Synchronous Measurement of Stability of Electron Beam, X-Ray Beam, Ground and Cavity Voltage", DIPAC'11

FREE AND OPEN SOURCE SOFTWARE AT CERN: INTEGRATION OF DRIVERS IN THE LINUX KERNEL

Juan David González Cobas, Samuel Iglesias Gonsálvez,
Julian Howard Lewis, Javier Serrano, Manohar Vanga, CERN, Geneva, Switzerland
Emilio G. Cota, Columbia University, NY (formerly at CERN), U.S.A.
Alessandro Rubini, Federico Vaga, University of Pavia, Italy

Abstract

Most device drivers written for accelerator control systems suffer from a severe lack of portability due to the *ad hoc* nature of the code, often embodied with intimate knowledge of the particular machine it is deployed in. In this paper we challenge this practice by arguing for the opposite approach: development *in the open*, which in our case translates into the integration of our code within the Linux kernel. We make our case by describing the upstream merge effort of the `tsi148` driver, a critical (and complex) component of the control system. The encouraging results from this effort have then led us to follow the same approach with two more ambitious projects, currently in the works: Linux support for the upcoming FMC boards [1, 2] and a new I/O subsystem.

TSI148 DRIVER INTEGRATION IN THE KERNEL

Rationale

The VME bus is a central component of the Controls System at CERN. We rely on 1140 FECs (Front End Computers), 710 of which are VME crates with SBC (Single Board Computers). A process of renovation is in course, involving the migration from

- CES RIO2/RIO3 SBCs with PowerPC CPUs runing LynxOS (around 605 crates by August 2011), to
- MEN-A20 SBCs with Intel CPUs running real-time Linux (around 105 by August 2011).

The MEN-A20 SBC incorporates a TSI148 VME-to-PCI-X bridge chip. To support the functionalities required, and for maximum backward compatibility with the legacy CES API, a device driver was developed by CERN's BE/CO group in spring 2009. Clearly, this is a critical component of the controls system: every VME device relies on the provided programming interface to the VME bus. The CERN-developed driver offers, therefore, a CES compatibility API to facilitate the transition during the renovation process and a new API more in line with what is common practice in the Linux kernel.

Work on the inclusion of the driver in the Linux kernel started in late 2010 and is currently nearing completion. This effort, along with its motivations and consequences, is documented in the remainder of this Section.

Benefits

There are many reasons that make the insertion of code in the mainline kernel a desirable target.

- Smoother maintenance in the (frequent) event of kernel API changes. [3]
- Very strict process of peer review of the code by knowledgeable and specialised maintainers.
- Widespread distribution of the code base, which can then be enhanced and get contributions by researchers working on similar problems.
- Input from the topmost experts in the field.
- Best practice and use of bleeding-edge tools selected by experienced programmers, *e.g.* git [4], sparse [5] and Coccinelle [6].
- Avoidance of suboptimal, *ad hoc* solutions in favour of the best ones from the technical point of view.
- Contributing back in return to the many benefits the FOSS community gives us.
- Being able to drive a critical hardware component with software in the vanilla kernel, with no local, idiosyncratic modifications.

It is worth noting that the importance of the last point only became apparent once the merge effort was well underway. In our control system, renovated FECs are *diskless* machines that boot a custom *rt* kernel [11], carefully configured and patched to match local needs. The upstream merge of the *tsi148* driver allows us to deploy off-the-shelf kernels packaged by linux distributions, thus largely reducing the kernel maintenance burden.

Caveats

Some of the caveats herewith enumerated will be elaborated further in the description of the integration process, but we summarize here the most important ones.

- It is hard, frustrating at times. One should be ready for the peculiar culture of the Linux Kernel Mailing List.
- Design, APIs and coding practice that are customary or simply acceptable locally must be adapted or rebuilt to comply with the strict standards imposed by the Linux kernel developers. The morale is that the end product will most certainly be different to (and better than) the original.

- One must be prepared to compromise. The most practical, short-lived solution might not be the technically perfect one kernel maintainers aim at.
- Maintainers are occasionally hard to deal with.
- The review process may be long; several iterations are usually necessary when merging significant changes.
- Small, incremental changes are more likely to be accepted than big, hard-to-digest ones. The rule of thumb is to "Separate *logical changes* into a single patch file." [7]
- Having a good history of prior contributions gives more respectability and ease of acceptance: the system is based on meritocracy.

The Process

Submission of source code for acceptance in the kernel is done by means of patches subject to a process of peer review before acceptance. The reviews are sometimes daunting, and much editing and re-submissions can result. This process of strict code review is another good practice our team adopted for its development process, a practice that has proved very beneficial.

Our fist attempt of integration of the `tsi148` driver was initiated in late 2010 by one of the authors (Emilio G. Cota).

By that time, a `tsi148` driver had recently been admitted in the `staging` [8] area of the kernel, its maintainer being Martyn Welch. This driver brought a tentative support for the VME bus that covered two VME-to-PCI bridges:

- The Tundra Universe chips, with work derived from VMELinux by John Huggins and Michael Wyrick.
- The Tundra TSI148 chip, inspired in the above.

The set of patches initially submitted provided improvements in several areas, esp. in orthodox implementation of the Linux bus/device model concepts. Acceptance by the maintainer was partial, and modifications related to the device model implementation remained controversial and not accepted.

In 2011, another of this paper's authors, Manohar Vanga, took over the submission effort. Some bugs in the staging driver were fixed, and the bulk of modifications concerning the device model were partially acknowledged. This led to several review iterations; at the time of this writing, the definitive submission and acceptance of the last set of patches concerning the core device model is in its final stage, and will be applied by the corresponding kernel maintainer in the next merge window.

Although the whole set of patches concerning bug fixes and device model of the `tsi148` VME bridge driver is finally accepted, there is still a long way before reaching the final desideratum, *i.e.,* driving our `tsi148` devices with stock software from the mainline kernel tree.

- The outstanding goal should be now getting the VME driver out of the `./staging/` tree. For this to happen, the overall quality of the code must improve significantly. [8]
- There are API incompatibilities with the driver currently used at CERN; this implies that a transient kernel module must adapt interfaces if driver code has to remain untouched.

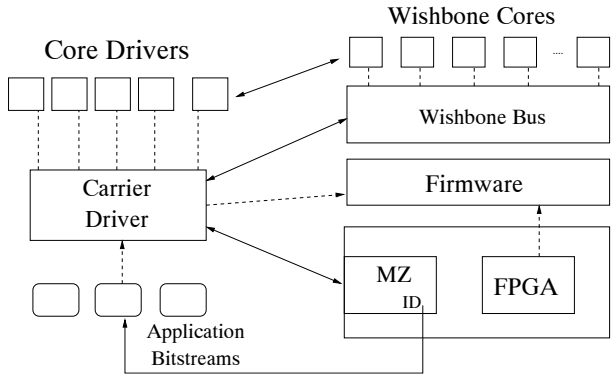


Figure 1: Wishbone bus driver architecture.

DRIVERS FOR THE FMC FAMILY

A family of carrier/mezzanine modules compliant with the ANSI VITA 57 FMC standard [2] is described in [1]. This kit of boards, developed by the BE/CO Hardware and Timing section at CERN is another case in point for the benefits of a Linux kernel-centric approach. The hardware concept and architecture are described in the aforementioned paper [1]; the key point is that the mezzanine provides basic circuitry, and the core of the application logic is implemented in the FPGA of the carrier board. The cores inside this FPGA are interconnected via a Wishbone [9] bus.

From the software point of view, a particular instance of this family behaves like a PCI-to-Wishbone or VME-to-Wishbone bridge. The Wishbone bus interconnects a set of cores providing functionalities that are either common to the whole family of mezzanines, or specific to the application board actually plugged in the FMC slot. We show in figure 2 the block diagram of the FMC 100MS 14 bit ADC, a typical example of an FPGA application comprising cores for, among others

- Basic I^2C interfacing to the mezzanine board.
- Wishbone mastering.
- DMA access to DDR3 memory in the carrier board.
- Mezzanine-specific control logic (*e.g.* ADC programming/setup).
- Interrupt control.

The consequence of this modular design of the application FPGA is that the device driver architecture reflects the structure of this set of cores (see figure 1). The driver for the carrier board performs essentially the following basic functions

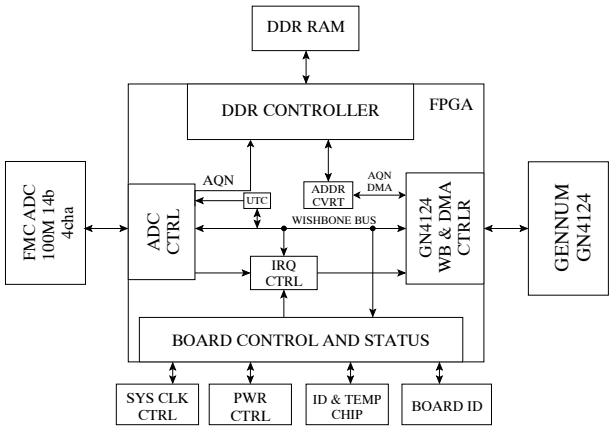


Figure 2: Block diagram of the FMC slow ADC application.

- Identify the carrier board and initialize it.
- Perform a basic identification of the mezzanine(s) installed in the FMC slot(s), and their configured applications.
- Load the application firmware into the carrier FPGA.
- Register a Wishbone bus with the kernel.
- Enumerate the cores in that firmware (to wit, the inner blocks in figure 2).
- Register the devices those cores implement and install the drivers associated to them.

The process is directly implied by how the Linux kernel device model is implemented as described in [10] or in chapter 14 of [12]. Conceptually, it amounts to the provision of a bus driver for Wishbone, plus a PCI-to-Wishbone bridge driver for the carrier board.

Modularity and reusability of cores and drivers are not the only rationale behind this design. Actually, the drivers for the FMC family are the second family of drivers planned to be integrated in the mainline kernel. Given that boards and their applications will be manufactured by external companies and available to the general public and not only to CERN, having their drivers and the Wishbone bus integrated upstream is the logical step to take. The Wishbone enumeration is made possible through the definition of an FPGA configuration space developed in [13], that will be the basis for the integration of all drivers of this family in the kernel.

DATA ACQUISITION DRIVERS: KERNEL FRAMEWORKS

The third family of drivers considered for integration upstream is less homogeneous than the FMC family. The BE/CO group supports a standard kit of hardware modules for data acquisition and general analog and digital I/O, some of whose characteristics can be compared in table 1. Linux drivers for these modules were developed in different stages of the renovation process, leading from legacy LynxOS drivers to Linux; therefore, there is a varying degree of adherence to standard practice among the Linux kernel developers, because of the LynxOS coding practice formerly prevalent.

Two needs arise here clearly, in the light of our intention of having our whole set of drivers incorporated upstream:

- Proceed to a more homogeneous, Linux-only code base.
- Provide a general framework for data acquisition and control devices.

Concerning the latter, there are two candidate Linux kernel frameworks: Comedi [14] and IIO [15]. Both are in `./staging/`, and careful analyses show that they prove insufficient for our needs. Therefore, the development of a more complete and acceptable framework for industrial data acquisition and analog/digital I/O becomes part of our effort of integrating our supported device drivers in the Linux kernel. This effort is motivated both by the requirements of legacy drivers and by the requirements that the FMC family will impose in driver design, esp. in the area of operating system interface.

Such a framework, codenamed `zio`, intends to cover all the relevant aspects of data acquisition devices, far beyond the existing frameworks, such as

- Digital and analog input and output.
- One-shot and streaming (buffered) data acquisition or waveform play.
- Resolution.
- Sampling rate.
- Buffer management and timing for streaming conversion.
- Support for DMA.
- Calibration, offset and gain.
- Bit grouping in digital I/O.
- Timestamping.
- Triggering of acquisition/output.
- Clean design conforming to Linux kernel practice.

At the time of this writing, prototype versions of the framework software are under development phase [16].

FORTHCOMING STRATEGY

The strategy for inclusion of our driver kit in the kernel relies now on a handful of key milestones

1. Initiate upstream integration of the Wishbone bus driver.
2. Make the in-tree and our local API for `tsi148` converge.
3. Get the VME driver out of `staging`.
4. Develop the `zio` framework as a competitive alternative to Comedi and IIO.
5. Initiate integration of the `sis33xx` drivers into the `zio` framework and the mainstream kernel.

Points 1, 2 and 4 are short-term priorities that should come out as quite straightforward.

Table 1: BE/CO Data Acquisition Modules

Module	Type	Channels	Resolution	Max. Speed	Bus
VMOD-12E8/16	Analog input	8/16ch	12b	15us/sample	VME/PCI
VD80	Analog input	16ch	16b	200kS/s	VME
SIS3300	Analog input	8ch	12/14b	100MS/s	VME
SIS3302	Analog input	8ch	16b	100MS/s	VME
SIS3320	Analog input	8ch	12b	250MS/s	VME
"Fast" FMC ADC	Analog input	4ch	14b	100Ms/s	VME/PCIe (Wishbone)
"Slow" FMC ADC	Analog input	8ch	16b	100kS/s	VME/PCIe (Wishbone)
CVORB V4	Analog output	16ch	16b	5us/sample	VME
VMOD-12A2/4	Analog output	2ch	12b	10us/sample	VME/PCI
CVORG	Analog output	2ch	14b	100 MS/s	VME
VMOD-TTL	Digital I/O	20ch	1b	n/a	VME/PCI
CVORA	Digital I/O	32ch	1–32b	100Mhz	VME

LESSONS LEARNED

The main lessons we got from this process, that we will have to take into account in the course of the forthcoming integrations, are

- Being there first gives competitive advantage. However technically sound a proposal might be, it is better to get it accepted when it is new, and not contending with other more entrenched positions.
- Getting input and enhancements from Linux peer developers is tremendously beneficial to improve the quality of the code, of the design and of the working environment (tools, policies and style being naturally given by what the Linux kernel developers have already tried and tested over many years).
- There is a change of thinking when it comes to development after a while in the kernel community. The mindset changes from thinking of the short term goals to thinking of solutions that can scale to a larger set of problems. One starts to think of more generic solutions rather than quickly hacking together the fastest solution to the problem at hand.

CONCLUSIONS

Contrary to what conventional assumptions dictate, low-level software and development need not be intimately linked to, nor closely shaped after, the peculiarities of a single control system. Developing with a wider scope in mind is possible, as our experience proves. But not only that: it results in technically superior, more scalable and maintainable solutions.

Last, but not least, peer review, advice and a bleeding-edge set of tools created by top level programmers contribute to the efficiency and quality of the development process; a priceless gift we also owe to the community of Linux kernel developers.

REFERENCES

[1] P. Alvarez, M. Cattin, J. H. Lewis, J. Serrano and T. Wlostowski, "FPGA Mezzanine Cards for CERNs Accelerator Control System", in ICALEPCS'09, p. 376, 2009.

[2] VME International Trade Association, "FPGA Mezzanine Card (FMC) Standard", `http://www.vita.com/`

[3] G. Kroah-Hartman, "The Linux Kernel Driver Interface", see Linux sources under `Documentation/stable_api_nonsense.txt`.

[4] L. Torvalds. "git: The Fast Version Control System", see `http://git-scm.com/`.

[5] L. Torvalds. "Sparse - a Semantic Parser for C", see `http://sparse.wiki.kernel.org/`.

[6] J.L. Lawall, J. Brunel, N. Palix, R.R. Hansen, H. Stuart, G. Muller, "WYSIWIB: A declarative approach to finding API protocols and bugs in Linux code,", in The 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 43-52, 2009.

[7] "Submitting Patches", see `linux-src/Documentation/SubmittingPatches`.

[8] G. Kroah-Hartman, "linux-staging tree created", announcement on the Linux Kernel Mailing List, June 2010. See `https://lkml.org/lkml/2008/6/10/329`

[9] OpenCORES, "Wishbone B4: WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores", see `http://opencores.org/opencores,wishbone`

[10] "Linux Kernel Device Model", see `linux-src/Documentation/driver-model/`.

[11] S. Rostedt and D.V. Hart, "Internals of the RT patch". In Linux Symposium, 2007.

[12] J. Corbet, A. Rubini, G. Kroah-Hartman, "Linux Device Drivers", 3rd edition. 2005.

[13] FPGA config space Open Hardware Project, see `http://www.ohwr.org/projects/fpga-config-space/`

[14] See `http://www.comedi.org/`

[15] See article in `http://lwn.net/Articles/339674/`

[16] See `git://gnudd.com/zio-beta.git`

THE CASE FOR SOFT-CPUS IN ACCELERATOR CONTROL SYSTEMS

W. W. Terpstra, GSI Helmholtz Centre for Heavy Ion Research GmbH, Darmstadt, Germany

Abstract

The steady improvements in Field Programmable Gate Array (FPGA) performance, size, and cost have driven their ever increasing use in industry, science, and IT. As FPGA sizes continue to increase, more and more devices and logic move from dedicated chips to FPGAs. For simple hardware components, the savings in board area and chip count are compelling. For logic involving dynamic memory and complicated control flow, the trade-off is not as clear.

Traditionally, this has been the domain of CPUs and software programming languages. In hardware designs already including an FPGA, it is tempting to remove the CPU and implement all logic in the FPGA. However, if that logic is then be implemented in the more constraining hardware description languages, it cannot be as easily debugged or traced, and typically requires significant FPGA area. For performance-critical tasks, this trade-off can make sense. However, for the myriad slower and dynamic tasks, software programming languages remain the better choice.

One great benefit of a CPU is that it can perform many tasks. Thus, by including a small "Soft-CPU" or softcore inside the FPGA, many low performance tasks can be aggregated into a single component. These tasks may then reuse existing software libraries and debugging techniques, while retaining ready access to the FPGA's internals.

This paper discusses requirements for using Soft-CPUs in this niche, especially for the FAIR project[1]. Several open-source alternatives will be compared and recommendations made for the best way to leverage a hybrid design.

BACKGROUND

FPGA chips allow developers to implement custom circuitry. A developer designs his desired circuit in a Hardware Description Language (HDL), and a compiler translates this into the underlying gates and wires required by the design. The target FPGA chip can then be programmed with the resulting bitstream. A given FPGA has a limited number of wires and gates and when a design is loaded, it consumes some of the *area* on the chip.

Developers organize their HDL project into components. A component corresponds roughly to a chip design, with internal logic and input/output pins. For example, a component might implement a memory chip with address and data pins. Each component can be instantiated multiple times, like placing several chips with the same specification. These instances can in turn be wired together to produce a larger component, akin to wiring several chips together on a card. The resulting amalgam component will have it's own input/output pins. This hierarchical composition results in a tree of nested component instances.

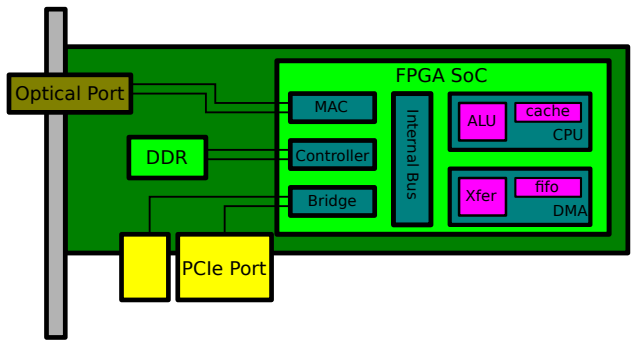


Figure 1: A SoC including a Soft-CPU.

Typically, the top-most component or root has its input/output pins mapped to the physical pins on the FPGA. The FPGA's pins are then physically wired to external chips mounted on the board and the connectors required by the type of device manufactured. For example, in the FAIR project, our control devices have FPGA pins connected to a DDR memory chip, an optical network transceiver, and a PCI express connector.

Within the FPGA, to ease interconnection of several components, developers often use a bus protocol. Just like the PCI express bus of a PC, this allows developers to plug together multiple components without needing to modify their designs. The resulting FPGA is called a System on Chip (SoC), because it is effectively an entire computer in a single chip. The key advantage of a bus is that it allows multiple logical connections. Where a simple wire between components only connects those two components, a bus allows n master components to control m slave components, resulting in nm logical connections. Figures 1 and 2 illustrate two SoC designs.

Within a SoC system, one can include a CPU. This CPU is typically a bus master which controls the slave devices on the SoC bus. As the CPU is not a physically distinct chip, and is implemented in HDL within the design, it is called a Soft-CPU or SoftCore.

This paper will cover: when it makes sense to include a Soft-CPU, what they require from the memory subsystem, an analysis of the best available Soft-CPUs, and our recommendations.

[1] A new international accelerator facility for antiproton and ion beams.

WHEN TO USE A SOFT-CPU

Soft-CPUs are often, but not always, a good thing to include in a SoC design. On a modern FPGA the area they consume is relatively minor; an LM32 Soft-CPU uses 2-4% of a EP2AGX125, depending on selected features. However, they typically impose a hefty memory cost, discussed later. Furthermore, they introduce another master running on the bus, which may not be desirable due to arbitration.

Comparison to Custom HDL

Any functionality one can implement in software, one can also implement directly in HDL; it's just more work. For example, `memcpy(dst, src, len)` run on a Soft-CPU can move data from one slave device to another. In HDL one could implement a DMA controller to perform the transfer. Broadly speaking, the advantage of HDL is raw speed and software is ease of development. More specifically, the strengths of a Soft-CPU are:

- ✓ Soft-CPUs can run traditional C/Fortran/Java code. They can re-use existing libraries and applications.
- ✓ A single Soft-CPU can implement many features with one component; one processor can run multiple programs. In HDL each feature requires more circuitry.
- ✓ Execution order is easier in software. In HDL a state machine tracks the current operation. In software this is implicit in the program's instruction pointer.
- ✓ Dynamic resource management is easier in software. Software systems have a stack and a heap. Hardware only has static (in the C/C++ sense) variables.
- ✓ Software debuggers can single-step programs and inspect/modify every variable. Hardware signal tapping captures execution after matching crude triggers and can only inspect new variables with a recompile.
- ✗ Software is *much* slower. Hardware is always parallel, executing many things at once. However, Soft-CPUs can leverage this parallelism with custom instructions.
- ✗ Soft-CPUs require a memory subsystem.
- ✗ Soft-CPUs require an additional developer toolchain.

It is important to understand that Soft-CPUs and custom HDL are not mutually exclusive; the above pro/con list presents a false dichotomy. One of the key strengths of a Soft-CPU compared to an external CPU is that it is directly connected to customizable hardware. The question is not whether or not to use custom HDL. When using an FPGA that is a given. The question is if any of the required functionality could benefit from a Soft-CPU. If a Soft-CPU is useful, then a hybrid design should be used.

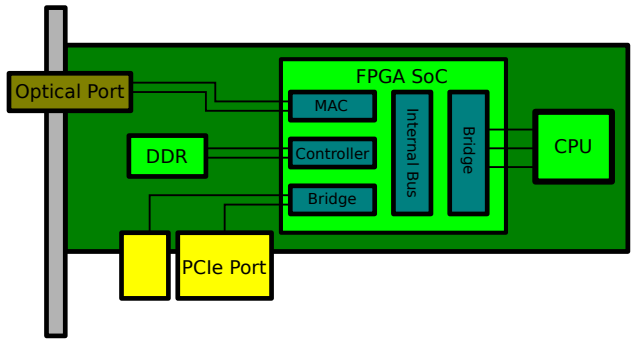


Figure 2: A SoC connected to an external CPU.

Comparison to an External CPU

If a CPU is required, instead of putting it inside the FPGA, one can attach it as an external chip. See, for example, Figure 2. All of the same trade-offs for custom HDL that apply to a Soft-CPU apply to an external CPU. However, Soft- and external CPUs have their own trade-offs.

The main advantage of an external CPU is performance. Soft-CPUs are as simple as possible to minimize the FPGA area required. Generally, their designs mimic RISC architectures from the mid 1980s. Modern superscalar CPUs are significantly more complex and much faster clock for clock. Furthermore, the fairly decent EP2AGX125 FPGA run an LM32 Soft-Core at 200MHz compared to the typical 3GHz clock rates of a modern CPU. Thus, an external CPU will dispatch over 30 times as many instructions as a Soft-CPU. Nevertheless, for a particular problem, custom hardware can be faster still and a Soft-CPU will be directly connected to that hardware.

Since the main advantage of an external CPU is raw performance, it will be run at a faster clock rate than the FPGA. Otherwise, one might as well use a Soft-CPU and save money. In this configuration, the trade-offs are:

- ✓ The Soft-CPU solution costs one less chip.
- ✗ An external CPU issues instructions $> 30\times$ faster.
- ✓ The Soft-CPU can use custom instructions.
- ✗ The external CPU runs a standard OS and toolchain.
- ✓ A Soft-CPU operates synchronously with the FPGA.
  - It is directly connected to the internal bus. There is no bridge, no variable latency.
  - It is in the same clock domain as the devices.

When money is no object, the main advantage of a Soft-CPU is deterministic execution speed; the Soft-CPU executes instructions at a known rate. In a hybrid design, it is possible for a custom HDL component to dispatch work to the Soft-CPU and be guaranteed to receive the result after a fixed number of cycles. Similarly, when the Soft-CPU pushes data to a device known to use 5 cycles, it doesn't

need to synchronize or check for completion. It just burns 4 cycles before reading the result. This tight integration makes it possible for a Soft-CPU to fill the not-easily-done-in-HDL gaps of a mostly custom HDL solution.

MEMORY ARCHITECTURE

The real cost of adding a Soft-CPU to a design is not FPGA area, but memory. The more complex the logic that the Soft-CPU executes, the larger the executable it must run. While it is true that a single Soft-CPU can implement many features in one component, the hidden cost is this increasing memory foot print.

Unfortunately, a larger memory footprint comes with decreased determinism. As the memory requirements grow, the memory must move further from the CPU. As the memory moves further from the CPU, the memory hierarchy deepens. Caches make an otherwise straight-forward program take an unpredictable amount of time. This weakens the greatest strength of a Soft-CPU. Unfortunately, if the problem requires more memory, this is unavoidable.

Certainly, one could imagine running a Soft-CPU with an SRAM chip for memory and no cache. An SRAM chip will take at least 4 cycles to provide each instruction to the Soft-CPU. FPGA internal cache memory could provide it in 1 cycle. If one runs the Soft-CPU with no cache or prefetch, it is exactly 4 times slower. With cache, it is at worst 4 times slower and at best full speed. Omitting the cache doesn't improve the deadline behaviour of the Soft-CPU; it just makes it (much) slower on average. Thus, when the Soft-CPU's memory footprint exceeds the memory available at one level in the memory hierarchy, that level becomes cache for the next, larger level.

The important message is that one size does not fit all problems. A good hybrid design uses the smallest amount of memory possible, and thus the least levels of cache.

Another factor which can impact determinism is the bus itself. If the Soft-CPU must share access to the memory with other devices, it may need to wait for those devices to finish their access. We strongly recommend that a hybrid design dedicate a memory port exclusively to the Soft-CPU's instruction bus. The data bus is much less frequently used (only on load/store instructions) and can share access with other devices as long as access is carefully scheduled.

MMU: Yes or No

A common feature found on modern CPUs is the Memory Management Unit (MMU). This introduces a level of indirect addressing which is needed by modern operating systems to implement inter-process memory protection. Some Soft-CPUs always include an MMU, some don't, and for some it is configurable. An MMU implementation is typically tightly integrated with the CPU cache system. Usually a Translation Lookaside Buffer (TLB) maps virtual addresses to physical addresses. When the TLB cannot find an entry needed, it must be refilled, introducing an additional source of non-deterministic delay. The trade-offs:

✓ Standard OS supported, meaning more compatability.

✓ Supports efficiently garbage collected languages.

✓ Fault isolation and memory error detection.

✗ TLB misses add non-deterministic execution delay.

✗ Memory required for page table management.

The main advantages of an MMU only apply to large software systems. Both a standard OS and a garbage collected runtime entail significant memory consumption. Furthermore, both of these scenarios make steep concessions in pause times. If one uses a Soft-CPU with an MMU, one has almost given up the main advantage of a Soft-CPU compared to an external CPU. Both the MMU and attendant cache system introduce execution pauses, as does a standard OS and/or garbage collected runtime. Aside from cost, little remains to recommend a Soft-CPU + MMU.

SOFT-CPU SHOWDOWN

To choose a Soft-CPU for use in the FAIR project, we made a list of requirements that candidate Soft-CPUs should fulfill. In order of decreasing importance:

- Open source HDL: port to many FPGA manufacturers, add required features, trace and debug problems.
- Toolchain support. No compiler, no point.
- Configurable memory subsystem.
- Mature and well documented.
- Size and Speed.
- Debugger support.

We only considered CPUs which met at least the first two requirements. The surviving candidates we investigated are summarized in Table 1.

RECOMMENDATIONS

Inspecting Table 1 makes pretty clear that the only reasonable options are the LEON3, LM32, and OpenRISC processors. All three processors are RISC and have similar area and performance. The other open source offerings are either incomplete, poorly documented, or too large for an FPGA (S1). Of the runner ups, the ZPU is probably the most interesting due to its smaller foot-print. However, it is significantly slower clock-for-clock than the three main candidates which issue most instructions in one cycle.

For a hybrid design, the LM32 is the clear winner. It has very clearly specified instruction timings and is designed to work without an MMU. It's memory subsystem is very configurable; the instruction bus can use either internal FPGA memory, an 1/2-way instruction cache, or direct bus access. This makes it easily accommodate designs

Table 1: Feature Breakdown of Available Open Source CPUs as Synthesized on an Arria2

CPU	Memory Subsystem	Documentation	Area	Speed	Debugger
LM32	optional(I+D cache)	Very Good [4, 5]	2000	200MHz	yes
LEON3	optional(I+D cache + MMU)	Excellent [1, 2, 3]	3000	150MHz	yes
OpenRISC	I+D cache + MMU	Adequate [6, 5]	3300	150MHz	yes
ZPU	none	Poor	1000	200MHz	no
ZET	none	Poor	3000	60MHz	no
S1	I+D cache + MMU	Sparc [3]	45000	-	-
CPU86	none	Non-existant	3800	100MHz	no
Plasma	none	Non-existant	-	-	no
Navre	none	Non-existant	1000	200MHz	no
pavr	none	Poor	2400	120MHz	no
aemb	none	Non-existant	1100	140MHz	no
openfire	none	Non-existant	1300	160MHz	no
pacoblaze	none	Poor	650	200MHz	no
yasep	none	Poor	-	-	no
dspuva16	none	Non-existant	340	250MHz	no

with different memory requirements. Finally, it is architecturally much simpler than a sparc. A hand-crafted operating system can easily fit in under 1kB of memory, or it can just be used directly as a micro-controller.

Both the LEON3 and the OpenRISC are well suited to running a full Linux operating system. However, the only thing which recommends them over using a much faster external CPU is cost. When running linux, the operating system, MMU, and cache all conspire to eliminate the benefit of a direct connection to the FPGA bus. An SoC bridged over PCIe to an external CPU would likely meet tighter real-time deadlines.

In conclusion, given the low area cost of a Soft-CPU and the many potential benefits, it seems wise to plan on including a Soft-CPU in most SoC designs. As long as the executable code can fit in FPGA memory, the design impact is low. At the GSI, we have now used the LM32 in several designs and found it very useful not only in offloading low-priority tasks from custom HDL, but also in debugging attached hardware devices.

APPENDIX: LM32 FEATURES

A quick overview of our recommended Soft-CPU:

- 6-stage pipelined RISC architecture
- 32-bit data path and instructions
- 32 general-purpose registers and interrupts
- Optional instruction and data cache
- Wishbone [5] memory interfaces (instruction+data)
- Debug unit with breakpoints+watchpoints
- Can both be debugged over JTAG or internally

REFERENCES

[1] Aeroflex Gaisler, “GRLIB IP Library Users Manual”, 2010. www.gaisler.com/products/grlib/grlib.pdf

[2] Aeroflex Gaisler, “GRLIB IP Core Users Manual”, 2010. www.gaisler.com/products/grlib/grip.pdf

[3] SPARC Int’l, “The SPARC Architecture Manual Version 8”, Prentice-Hall, Inc., Upper Saddle River, NJ, 1992, www.sparc.com/standards/V8.pdf

[4] Lattice Semiconductor Corporation, “LatticeMico32 Processor Reference Manual”, August 2007, www.latticesemi.com/documents/doc20890x45.pdf

[5] R. Herveille, “WISHBONE System-on-Chip (SoC) Interconnection Architecturefor Portable IP Cores”, 2010, cdn.opencores.org/downloads/wbspec_b4.pdf

[6] D. Lampret et al., “OpenRISC 1000 Architecture Manual”, April 2006, opencores.org/svnget,or1k?file=/trunk/docs/openrisc_arch.pdf

THE FAIR TIMING MASTER: A DISCUSSION OF PERFORMANCE REQUIREMENTS AND ARCHITICTURES FOR A HIGH-PRECISION TIMING SYSTEM

M. Kreider, GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany
Hochschule Darmstadt, Darmstadt, Germany
Glyndŵr University, Wrexham, UK

Abstract

Production chains in a particle accelerator are complex structures with many interdependencies and multiple paths to consider. This ranges from system initialisation and synchronisation of numerous machines to interlock handling and appropriate contingency measures like beam dump scenarios. The FAIR facility will employ WhiteRabbit, a time based system which delivers an instruction and a corresponding execution time to a machine. In order to meet the deadlines in any given production chain, instructions need to be sent out ahead of time. For this purpose, code execution and message delivery times need to be known in advance. The FAIR Timing Master needs to be reliably capable of satisfying these timing requirements as well as being fault tolerant. Event sequences of recorded production chains indicate that low reaction times to internal and external events and fast, parallel execution are required. This suggests a slim architecture, especially devised for this purpose. Using the thread model of an OS or other high level programs on a generic CPU would be counterproductive when trying to achieve deterministic processing times. This paper deals with the analysis of said requirements as well as a comparison of known processor and virtual machine architectures and the possibilities of parallelisation in programmable hardware. In addition, existing proposals at GSI will be checked against these findings. The final goal will be to determine the best instruction set for modeling any given production chain and devising a suitable architecture to execute these models.

INTRODUCTION

In an accelerator, machines need to execute their actions at predefined moment, as part of an overall master plan. Magnets need to be run through current ramps, kicker assemblies need to be triggered at the right moment for beam transfer between accelerator rings, and similar matters. Most accelerators use some form of middle ware to convert physical scenarios into the necessary machine commands, like the production of a beam with a given energy and isotope. In theory, it would be possible to compute all necessary instructions and their times of execution in advance and then just go with this program. For such a scenario one would only need a very simple timing control, executing this precomputed value table.

However, more flexibility is needed in a real world scenario, since circumstances can change. From simply waiting for a machine to report ready status over changing demands from operators up to occurring interlocks and engaging machine protection circuits, all of these require changes to the planned scenario. Given a certain deadline, satisfying the timing requirements depends on several factors: The time to detect the condition and compose the message in the data master, transmission time through the distribution network, composed of switch traversal and time on the line, and decoding time in the endpoint.

TIMING CONSTRAINTS

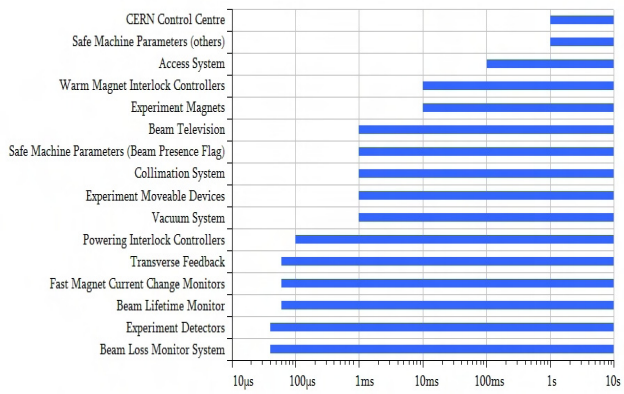


Figure 1: Reaction Times of CERN Interlock systems.[1]

The system timing constraints are to be deduced from hardware parameters like the minimum possible reaction time of a magnets power supply or the synchronisation requirements with RF systems and kicker assemblies as well as interlock systems, subdivided into beam protection, machine protection access control and operating. For all accelerators, interlock handling comes in several speed categories, depending on the kind of system to be monitored. from fastest to slowest these are: beam diagnostic equipment, magnet power supplies, vacuum system, temperature control, access control and finally operating.

Figure 1 shows a comparison between various systems employed at CERN's LHC. Interlock deadlines are spread here from 40μs over the low millisecond range up to several hundred milliseconds [2].

While many of these seem to require comparatively slow reaction achievable with a Real Time Operating System (RTOS), the fastest reaction times possible are always desired. The reason is, the quicker the system reaction time,

[1] B.Todd, A Beam Interlock System for CERN High Energy Accelerators

the less beam dump scenarios are to be expected. This means less ionisation of beam guides and Faraday cups, reducing radiation emmanating from the facility and therefore cooldown time in the beginning of maintenance shutdowns. It also causes less damage to beam diagnostic equipment. Availability for hands-on-maintenance goes with an overall decrease in maintenance time and cost. The most critical machine protection system shown here is the quench protection system for superconducting magnets. As to underline the importance of this system, the most prominent example is the 2008 incident at CERN. A quench led to the explosive discharge of about two tons of helium from a magnet housing into the LHC tunnel, damaging the equipment so severely, that the LHC was shut down for repairs until the end of 2009.

As an example from the planned GSI/FAIR facility, the shortest period to be served by the timing system is about 3ns for the kickers. This is obviously too fast for a reaction over the timing and controls network, but there is mixed approach to meet requirements and enhance system safety at the same time. The kicker timing depends on the RF frequency currently present on the accelerator ring, and this frequency is also sampled at the data master. Phase is therefore known to the whole system. After the kicker signals readiness, the event is scheduled to an absolute time 3ns after a future RF edge. The edge is chosen in a way to allow for transmission time of the control message trough the timing network plus a safety margin. Because transmission and processing time is somewhere around 100 μs, the kicker can safely be assumed to be unchanging in the interim. Until now, the kicker at GSI was triggered on a local basis without regard for other circumstances in the system, while said approach would shift back control to the timing master.

TRANSMISSION TIME

For an example of transmission and decoding times over a WR [1] network, we took a closer look at the future timing system of two accelerators, GSI/FAIR and CERN. The following estimates from the White Rabbit Robustness evaluation [3] show the time for transport of command messages on the network.

Table 1: Elements of Control Message Frame Delivery Delay Estimation [2]

Name	**Value** μs Min	**Value** μs Max
Eth Frame TX Delay	0	$(13 + B_{tx} t_{FECpck})$
Switch Routing Delay	0	13
Link Delay	$5[\frac{\mu s}{km}]$	$5\,[\frac{\mu s}{km}]$
Eth Frame RX delay	t_{FECpck}	t_{FECpck}
FEC Encoding	2	2
FEC Decoding	2	2

Both GSI/FAIR and CERN will employ a tree topology with three layers of WR switches between master and end-points. The main difference between systems in the calculation is in the estimated length of fiber links used. The value is 2 km for GSI/FAIR and 10 km for CERN. The following values are calculated under the assumption that switches have a cut-through option to preempt command messages.

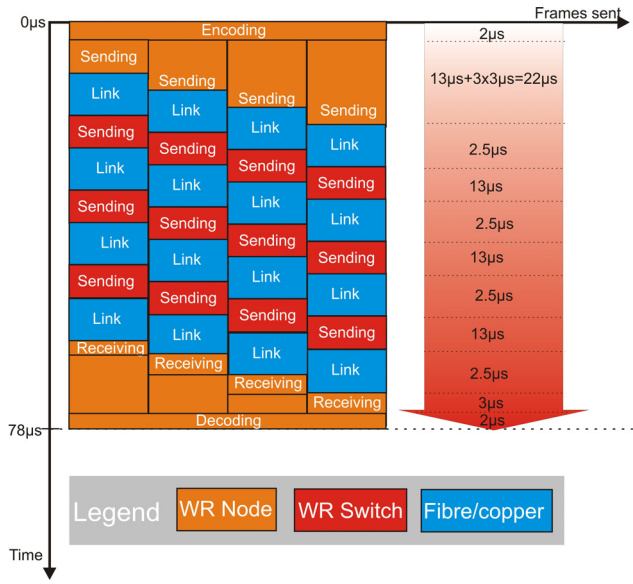


Figure 2: Delivery Delay of Control Message using cut-trough.[2]

Table 2: Control Message Delivery Delay

Control Message size	**Delay**	
	GSI	CERN
500 bytes	78μs	118μs
1500 bytes	102μs	142μs
5000 bytes	162μs	202μs

The figures show that for interlock servicing, it would be advantageous to connect time critical devices to the upper switch layers in order to reduce transmission time. For fastest respone, connection to the topmost layer would reduce latency by about two thirds. From a practical point of view however, since the tree is fanning out, this will of course be limited by number of ports available on the first switch layer or even the timing data master.

MODELING THE ACCELERATOR

Command messages sent over the timing network supply event codes to the machines paired with execution times. The machines are preprogrammed with a certain reaction upon reception, which should also be done by the same means. The accelerator has sequential processes with synchronisation points. While the sequences are simple in themselves, their interdependencies can be quite complex,

[2] M.Lipinski, C. Prados, White Rabbit Robustness

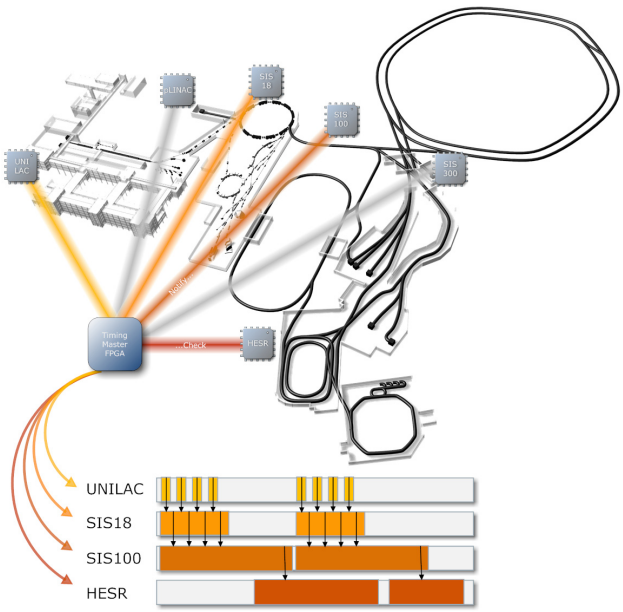


Figure 3: Machine sequences in a production chain.

depending both on external events like interlocks and internal synchronisation. Sequence architecture include jumps, conditional branching, IO handling, loops, real time clock waits and fast synchronisation between sequences. The sequences must be exchangable in parameters or completely during run time. For the GSI/FAIR accelerator, about 20 of these sequences are expected to run in parallel, 32 are aimed for to be future-proof.

PROCESSING TIME

CPU

With increasing complexity and number of processes, keeping code execution and synchronisation deterministic becomes ever more challenging. Fast modern CPUs, especially multicore architectures, offer vast processing power. Their drawback is that they require a managing OS to unlock most of their advanced features, which have worst case interrupt service latencies in the low millisecond range, making it an inapt choice for a Timing Master architecture. Real time Operating Systems are deterministic and can offer ISR latencies in the low microseconds range and most them are made for embedded CPUs (MCU). There are no RTOS systems targeting Hi-End-Multicore architectures available today.

Table 3 shows the latency measurements for a PowerPC 604 CPU (300MHz) on a MVME2306 board [4]. The discussed timing constraints could possibly be achieved with an MCU running an RTOS system, but as shown in Table 3, jitter is in the range of several microseconds. There are also strong peaks under load (which would be common, since the RTOS has to handle all machines and IO sources in the system) which go as high as 200μs. This disqualifies the solution for all fast interlock systems. We must also bear in mind that transmission time through the timing network ranging between 80-200μs must be added to the total reaction time, restricting the use even further. This leads to the assumption that an MCU with RTOS would probably be capable of running the required number of machine agents, but as figures show, could hardly cope with timing requirements for IO service requests.

Table 3: RTOS Latency Measurement Results[3] Times in μs

	Interrupt Latency		Context Switching	
	max	**avg** $\pm$	**max**	**avg** $\pm$
Idle System				
RTL	13.5	(1.7 $\pm$ 0.2)	33.1	(8.7 $\pm$ 0.5)
RTEMS1	14.9	(1.3 $\pm$ 0.1)	16.9	(2.3 $\pm$ 0.1)
RTEMS	15.1	(1.3 $\pm$ 0.1)	16.4	(2.2 $\pm$ 0.1)
vxWorks	13.1	(2.0 $\pm$ 0.2)	19.0	(3.1 $\pm$ 0.3)
Loaded System				
RTL	196.8	(2.1 $\pm$ 3.3)	193.9	(11.2 $\pm$ 4.5)
RTEMS1	19.2	(2.4 $\pm$ 1.7)	213.0	(10.4 $\pm$ 12.7)
RTEMS	20.5	(2.9 $\pm$ 1.8)	51.3	(3.7 $\pm$ 2.0)
vxWorks	25.2	(2.9 $\pm$ 1.5)	38.8	(9.5 $\pm$ 3.2)

FPGA Based Soft-CPU

This leaves programmable hardware for investigation. Field Programmable Gate Arrays (FPGA) can be configured by Hardware Description Languages (HDL) to function like every digital circuit from a simple FlipFlop over a counter to a full blown embedded CPU. The major advantage is in their huge degree of flexibility. There are several open source Soft-CPUs (Soft-CPU) as well as a couple commercial ones available today. Soft-CPUs in FPGAs are a fast, lightweight alternative to real hardware CPUs. Their most interesting aspect is that it is possible to have enough instances inside an FPGA to actually assign one Soft-CPU to each task. They also do not accumulate wait times for bus arbitration and memory access in operation, because they do not need to share their memory resources. A modern FPGA not only hosts logic units, but memory cells as well. Creating an instance of a memory controller template with matching RAM block for each Soft-CPU inside the FPGA makes them independent of their peers. In addition, this RAM block is natively capable of Dual Port access, providing the capability for on the fly change of sequence programs. Due to the fact of the one Soft-CPU-per-task policy, sequence programs can be blocking instead of using ISR, saving the time consuming context change. Assuming 125MHz system frequency and therefore 8 ns period per cycle, response times to external IO are well below 0.5μs, beating all competition. Due to the given flexibility, building a complete custom Soft-CPU tailormade for the given scenario is possible. The major drawback of a custom processing unit, wielding its own instruction set, is the total lack of toolchains. Virtually everything in a toolchain

[3] T. Straumann, Open Source Real Time Operating Systems Overview

that makes a productive workflow possible, that is, assembler, linker, compiler, debugger etc must be created from scratch. There will most likely be no community support, since the application is very narrow. Since the scenario requires standard functionality like basic math, counters, loops and RTC support, it will be a lot of redundant work.

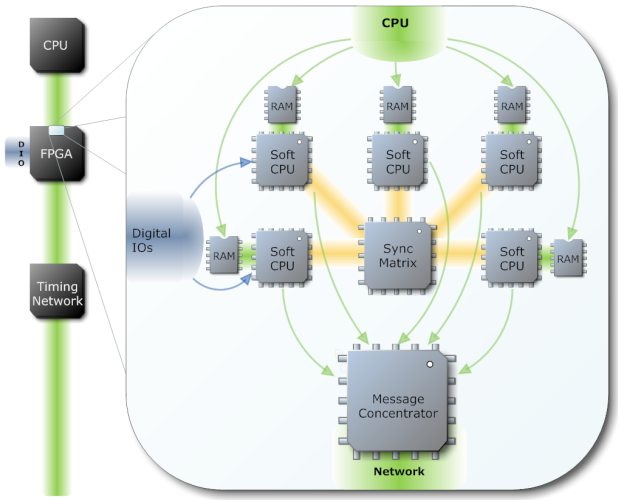


Figure 4: FPGA based Soft-CPU Cluster.

Soft-CPU Evaluation

A more promising approach is therefore to use a tried and tested open source soft CPU and enhance it with custom instructions. In our case, those are mainly ones used for quickly synchronising of soft CPUs.

Soft CPUs like the LM-32 are at around 2000 logic cells and therfore very lightweight [5]. Today's high end FPGAs range in the $>$ 250000 logic cells and can therefore easily instantiated several tens of those CPUs. According to early tests, the limiting factor is more likely to be the amount of RAM present in an FPGA than the number of logic units. Assuming around 2.5MB internal memory cells for a modern FPGA, this equals to about 80kB for each of the 32 soft-cores desired for the FAIR accelerator, more than enough to execute quite big programs. As an example, running a GNU debugger on system currently consumes around 1 kB memory.

The idea is then to create a Soft-CPU for each machine handler as well as one or more instances dedicated to IO signal and high level communication handling, as shown in figure 4. So instead of solving a complex scheduling problem in software to maintain deterministic system behavior, parallel tasks each get assigned their own Soft-CPU. Their synchronisation can be done by a custom HDL block, an $n * n$ matrix for synchronisation messages, making sync possible within less than five processor cycles ($<$ 40 ns) for checking of simple flag bits.

In the field of experimental electronics, a multi author situation should also be considered. It makes sense to let people working with the actual experiment write the software, which is another point in favor of existing toolchains.

CONCLUSION

In sight of time availabe to react to system interlocks and signals, it is evident that a pure CPU solution will not be able to satisfy the requirements. A pure custom hardware solution probably would satisfy the requirements for any given constraint, but prove difficult when it comes to follow changes at the operating level. Soft-CPUs are a good compromise between both worlds, with the added benefit of readily available development tools. To provide more IO speed and better scalability in face of system load, the choice falls to a multi core architecture from Soft-CPUs with dedicated internal memory. This setup shall be completed by a fast synchronisation mechanism and the possibility to dedicate Soft-CPU to specific sequence programs and IO handling.

OUTLOOK

The first steps will be the implementation of customized Soft-CPUs in a small scale test system in order to measure actual response times. An mechanism for safe on-the-fly change of the sequence programs in Soft-CPUs has to be devised, and several Soft-CPUs will be tested in a small cluster to demonstrate fast synchronisation between sequence programs. A prototype system is planned to be set up in 2012 to drive the first component to be comissioned to the FAIR extension of the GSI accelerator, a proton linear accelerator module.

REFERENCES

[1] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, G. Gaderer, "White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet", IEEE Precision Clock Synchronization for Measurement, Control and Communication 2009, pp1-5, Brescia, October 2009.

[2] T. Straumann, "Open source real-time operating systems overview", Proceedings of the 8th International Conference on Accelerator and Large Experimental Physics Control Systems, San Jose, USA, 2001.

[3] M. Lipinski, C. Prados, "White Rabbit Robustness", 2011 `http://www.ohwr.org/projects/white-rabbit/repository/changes/trunk/documentation/specifications/robustness/robustness.pdf`, last visited 30.09.2011.

[4] B. Todd, "A beam interlock system for CERN high energy accelerators", Ph. D. thesis, Brunel University West London, 2006.

[5] W.W. Terpstra, "The Case for Soft-CPUs in Accelerator Control Systems", 2011, ICALEPS'11, THCHMUST05.

MODERN SYSTEM ARCHITECTURES IN EMBEDDED SYSTEMS

T.Korhonen, PSI, Villigen, Switzerland

Abstract

Several new technologies are making their way also in embedded systems. In addition to the FPGA technology which has become commonplace, multi-core CPUs and I/O virtualization (among others) are being introduced to the embedded systems. In this paper we review the trends and discuss how to take advantage of these features in control systems. Some potential application examples are discussed.

THE DOMINANT TRENDS

Many of the technical advances in the computing industry are driven by the needs of the applications in the general information technology, Internet and the web, telecommunication and the entertainment industry. Common for these markets are that the data volumes are increasing, as well as need for of data communication bandwidth and processing power. To respond to these demands, the computer and electronics industries have come up with several technology solutions.

The embedded and control system applications are in some areas in the leading edge of technology, but in other areas the technology used in the general computing is coming into the applications with a delay. This delay is to a large extent due to the nature of the embedded market: the application life cycle is much longer than in the general computing market.

The general trends of the computing market are not essentially new but have practically always been to a) provide more computing power and communication bandwidth and b) use the hardware investment in the most efficient way. In the next chapters we take a look at these solutions and what they could bring to the field of control systems. This is not always obvious and it is a time-consuming process to implement this into the embedded infrastructure.

NEED FOR SPEED

Computers are never fast enough. Each advance in the processor technology has taken us a step further, but as soon as a solution has become available, a number of applications have arisen that use up all the available power and require even more.

As increasing speed just by making the CPU clock rates higher has become impractical, the major method is now increasing the parallelism. This is happening in a number of ways, most apparent is putting multiple processors in one unit. However, this is not the only way to implement parallelism.

Speed by Parallelization

Starting at the lowest level of parallelization, FPGAs have been around for some time and have become very common in embedded applications, too. The recent applications have started to use them as processing engines instead of glue logic. An FPGA can be the ultimate tool for fine-grained parallel processing if the tasks in hand match the model. The downside of using FPGAs is that the application development is work-intensive, as the abstraction level of the tools is low. To implement an application in a FPGA the problem needs to be well constrained. FPGA is also a fairly expensive solution in many cases.

In the processor world the obvious way of introducing parallelism is to increase the number of processing units and divide the computing tasks between them, Multi-core CPUs have emerged in the recent years, being now the standard in server and desktop machines but have also recently entered the embedded market. There are essentially two different directions in multicore implementations: having a number of identical cores (typical in desktop computers) or having a mix of general and special purpose processors on the same chip, like the TI OMAP[1], which combines a (dual-core) ARM CPU,DSP and GPU on the same chip. This kind of heterogeneous multicore processors could be very interesting in applications that need floating-point and mathematical operations in the embedded level.

As an example of a multicore CPU targeted for the embedded market, the QorIQ SoC series by Freescale [2] is displayed in Fig. 1. In addition to a variable number of processor cores, it also integrates specialized function accelerators on the same chip, for instance a TCP packet accelerator that implements a part of the TCP packet processing in hardware. It also integrates Gigabit Ethernet transceivers and three PCI express endpoints plus a number of specialized function controllers like an encryption engine.

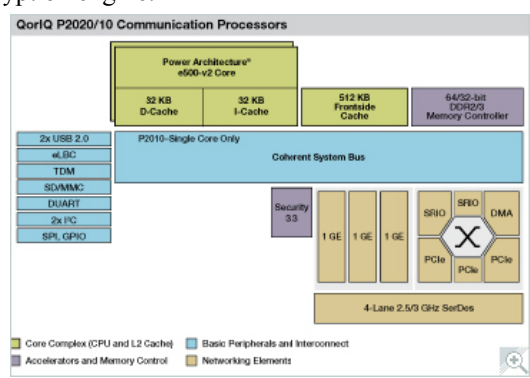


Figure 1. Freescale QorIQ® P2020 block diagram

Interconnects

To make advantage of the multiple processing units, the units need to communicate efficiently with each other. This has traditionally been implemented with multidrop buses. However, a bus scales very badly with the increasing number of processing units. For this reason, the

interconnects are more and more going to the direction of serial point-to-point links. In dedicated applications these links have been used already a long time. General-purpose interconnect protocols are now being included directly in processors so it makes a lot of sense to use them in embedded applications, too. The most prominent interconnect protocols are Ethernet (obviously) and PCI Express. The other protocols cater for specialized needs and what will happen with them in the long term is not that obvious. The two prominent protocols are however likely to have a large share in applications and be widely supported long into the future.

Controls Applications

A typical embedded application consists of tasks for I/O and some level of processing. As the processing power in the low level has grown, more and more applications have migrated to the low level. This has several advantages, as the data can be pre-processed right down at the source and only the relevant data needs to be passed on to the higher level. To achieve this in a low level embedded processor, there are a number of options. One can implement applications in a FPGA and take advantage of the real parallelism possibilities and very hard real time capabilities. This can however be very labour-intensive.

For the processing with a CPU, the obvious way is to try to take advantage of the multiple cores. A relatively simple way is to take advantage of a symmetric multiprocessing (SMP) OS and let it take care of distributing the load between processors. This can work quite well when the workload consists of different tasks that have little or no relation to each other. However, for embedded applications this is often not the case. The other possibility is then to dedicate computing cores for different tasks, or define certain tasks to run on different cores. This can help in management and partitioning of the workload.

The SoCs that provide different specialized cores can also have interesting possibilities. For instance the OMAP processors with their special processors could be efficient when the workload is a mixture of heavy floating point and matrix operations plus running general purpose tasks (I/O handling, network tasks etc., for which a general purpose core like an ARM is well optimized. Dedicating coprocessors for different tasks would also enable applications where a tight connection between a modelling environment and an embedded controller is useful. The modelling can happen in an workstation and the modelled application can be run on a dedicated core, where it would not need to disturb the general purpose tasks, and restarted as wanted.

Using GPUs as coprocessors for doing intensive matrix manipulations is also a possibility that is becoming more and more common.

QUEST FOR EFFICIENCY

Even if the price of computing equipment has come down, it still always costs too much. One of the long-standing ideas (and implementations) of taking the most benefit of the installed equipment has been to share the computing resources between several applications. Instead of each service running on its own machine, sharing the time of the hardware by running several things in parallel can improve the utilization of the hardware.

Virtualization

However, even more than optimizing the use of the hardware the problems facing IT departments now are physical space in the data center, power/cooling costs, system maintenance and management. The performance of servers has also increased to a point where there is a large amount of idle capacity. With virtualization, IT departments can utilize that spare capacity rather than adding a new physical server to support a new environment.

Virtualization is traditionally implemented with the help of a hypervisor that controls and shares the computing resources at a low level. The hypervisor is usually a thin software layer below the guest operating system. As the number of cores in CPUs increases, the task of a hypervisor gets more and more demanding and it self starts to become a bottleneck for the effective load sharing. Thus the functions of a hypervisor are increasingly being implemented in hardware. The need is most obvious and pressing in the area of I/O.

I/O Virtualization

The concept of I/O virtualization [3] is fairly recent and its meaning may not be immediately obvious. What this means is the implementation of the tasks of a software hypervisor in hardware to improve the efficiency. When many guest systems want to share the access to I/O modules the supervisor becomes a bottleneck when all the accesses have to be arbitrated. For instance, the SR-IOV standard within PCI express defines so called virtual functions in the devices. The guest systems can be given direct access to the hardware and the need for data manipulation by the hypervisor is eliminated.

SHARE AND CONQUER

Being a discipline with long development cycles, most of the software infrastructure in the control systems field is still oriented towards the single-processor model. Only recently activities to address have been initiated; some of them presented also in this conference. For instance the handling of timing of real-time processes: in a single-core system

The rise of open-source software has had a big impact on the way systems are built. Direct access to the source code has enabled implementation of functionality that was previously an area where only proprietary solutions existed. In addition, several working groups are defining open standards to make the utilization of multi-core systems easier. A notable standardization body is the Multicore Association [4], which has several working groups working on standardization of APIs in different

areas. For instance MCAPI working group provides a standardized API for communication and synchronization between closely distributed cores and/or processors in embedded systems. The working group has formed two subgroups. One is working on 'zero copy' functionality, including bidirectional interaction between 'application and application' using shared memory and bidirectional interaction between 'application and driver'. A second subgroup is focused on interoperability.

The MTAPI™ working group is charged with creating an industry-standard specification for an API that supports the task coordination on embedded parallel systems.

The Tools Infrastructure Working Group's key objective is to improve the interoperability between the different tool solutions for the development of embedded multicore systems, has been predominantly continuing its work on the Common Trace Format (CTF) specification. The goal of this work is seamless correlation of data coming from different types of tracers (e.g. software instrumentation, hardware instruction/data trace) as well as divergent execution contexts on multiple cores or systems.

PUTTING IT TOGETHER

Case for Parallelization and Multi-Core

There is no other way than going to multi-core – practically all modern CPUs have more than one core and the number will increase. However, the (real-time) control system software that runs on embedded processors has been written using single-core machines and while one can introduce operating systems that provide SMP support out of the box, just letting the OS to do the task assignment will cause some surprises in how the system behaves. The timing of tasks will inevitably change if tasks that have been assumed to be sequentially scheduled, are running truly parallel in separate cores.

Having said that, there are several cases where having multiple cores available can bring true advantages.

Data Streaming Application

One of the applications we are working on is a processing system for a low-level RF controller. The basic structure of the system is as shown in Fig. 2. The system collects data from several ADCs, processes them and does feedback of the RF system. The platform to do this is the IFC_1210 board that has recently been developed in collaboration between PSI and IOxOS SA [5]. The board is based on a PCI express communication infrastructure, on which several processing units are connected. The board has a powerful Virtex-6 "Central" FPGA, a dual-core Freescale P2020 QorIQ SoC CPU unit. The board in in VME format, to be compatible with the large existing installation base at PSI, but with an innovative use of the P0 connector allows PCI express links to be routed through the backplane, allowing us to create a multi-board system by connecting several boards together. The VME bus can be used as a control plane interface, but the system is not limited by the bus speed.

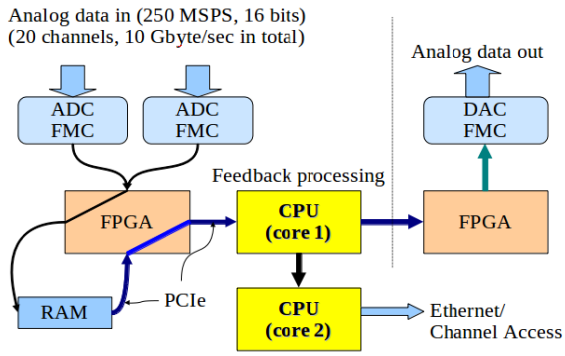


Figure 2. A data streaming and processing application for low-level RF control

The bulk of the raw data processing in this system takes place in the FPGA, however the control algorithms are at least in the first stage planned to be run on the general purpose PowerPC cores, that are much more suitable for interactive development. This makes the prototyping cycle much shorter. FPGAs do not easily lend them for scripting, even with the most modern tools.

The idea here is to take advantage of the multi-core CPU by clear partitioning of the tasks. At least in the development stage one could even take advantage of virtualization in the sense that the algorithms could run on a desktop machine which is connected to the field hardware via a long-distance PCI express link. The plan is to accelerate the development cycle by supporting direct access from modelling tools for developing and testing the feedback algorithms. The preferred tool for this is Matlab/Simulink. We plan to integrate it with the IFC_1201 board that we can directly download code from the tool to the system under test.

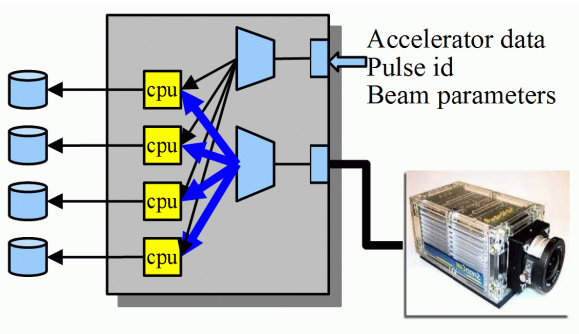


Figure 3. Striping data acquisition

Striping Data Acquisition

One conceivable application could be high-speed acquisition of data, e.g., images from a high-speed sensor to be processed in parallel. When the data cannot be processed fast enough by a single unit, it can be streamed to multiple units and the results stored in parallel (see Fig. 3). Many experiments in physics and X-ray FELs could fit this kind of model where each "event" is in principle an individual unit. If the imaging device would support I/O virtualization, the load could be effectively shared between multiple units. Usually the sensor data

needs also to be merged with data from the accelerator (e.g., pulse number, beam characteristics, etc.) to enable later analysis of the data.

Case for Virtualization

For controls applications, configuration management is probably the most compelling reason why one would consider virtualization.

The trend has been to put the computing closer to the equipment. With the fast interconnects and protocols available, one could think of revising the trend. It might be feasible to put the computing infrastructure together in a server room instead of spreading it out on the field. This would have the advantages that the management becomes easier, granularity of allocating processing power to the applications can be improved: not all I/O processors need to be the dimensioned according to the highest requirement, and on the other hand more power can be allocated where it is needed. In a sense this has already been happening since the EPICS software has allowed us to run the IOC core software on a server machine. Moving from the "soft IOC" would involve adding the direct (virtualized) I/O to the system and one could even implement real-time systems on a remote server. Using PCI express as the interface protocol, the infrastructure for device drivers would be easily usable in such an environment.

CONCLUSIONS

The advances in computing sometimes enter the embedded world with a delay. It is also often not obvious how to take advantage of the technologies that have been targeted to a different domain. However, after all the requirements are not that different, and by embracing the technology they can bring a lot of benefits to the embedded applications.

REFERENCES

[1] OMAP Application Processors; http://www.ti.com/omap.
[2] Freescale Corporation, www.freescale.com
[3] I/O Virtualization specifications. http://www.pcisig.com/specifications/iov/.
[4] Multicore association, www.multicore-association.org
[5] IOxOS SA, www.ioxos.ch

AN ERLANG-BASED FRONT END FRAMEWORK FOR ACCELERATOR CONTROLS

Dennis J. Nicklaus, Charlie Briegel, Jerry Firebaugh, Charlie King, Richard Neswold, Ron Rechenmacher, Jianming You. Fermilab, Batavia, IL 60510, U.S.A.

Abstract

We have developed a new front-end framework for Fermilab's Acnet control system in Erlang [1]. Erlang is a functional programming language developed for real-time telecommunications applications. The primary task of the front-end software is to connect the control system with drivers collecting data from individual field bus devices. Erlang's concurrency and message passing support have proven well-suited for managing large numbers of independent Acnet client requests for front-end data. Other Erlang features which make it particularly well-suited for a front-end framework include fault-tolerance with process monitoring and restarting, real-time response, and the ability to change code in running systems. Erlang's interactive shell and dynamic typing make writing and running unit tests an easy part of the development process. Erlang includes mechanisms for distributing applications which we will use for deploying our framework to multiple front-ends, along with a configured set of device drivers. We've developed Erlang code to use Fermilab's TCLK event distribution clock and Erlang's interface to C/C++ allows hardware-specific driver access.

INTRODUCTION

Our front-end computers are responsible for acquiring control system signals from the hardware or field bus and making this data available to the Fermilab control system. The front-end framework is a software architecture that is deployed to provide common central functions to serve the variety of devices or bus systems that we acquire data from. The primary requirements of this front-end framework are:

- Communicate readings, settings, etc. via the Acnet protocol
- Enable a mapping between the a device's name in the control system and the hardware channel.
- Manage the connections between control system requests and the data channel.
- Check for and report device alarms.
- Respond to the timing system to ensure prompt collection
- High degree of reliability.

We began discussing developing a new framework as an alternate path forward, replacing our vxWorks-hosted, C-language MOOC framework, with a stated goal of running under the Linux operating system. Initially, we were expecting to develop the new framework in C++, but the Erlang language caught our attention and we soon realized it would be a great candidate for our project.

ABOUT ERLANG

Erlang is a functional programming language originally developed for telephony applications. It was built to support soft real-time systems with a high degree of reliability and fault-tolerance. Erlang is designed to be scalable to very large distributed systems.

Functional Programming

Functional programming emphasizes the application of functions to get results, as opposed to the more familiar imperative paradigm which emphasizes changing state and side-effects of subroutines. The elimination of side-effects in functional programming makes predicting and testing function results more straightforward.

Properties of Erlang include dynamic typing, single assignment, and extensive support for pattern matching in assignment. The initial hurdle that new Erlang developers must jump is figuring out how to program when you can't change the value of a variable once it has been assigned. As you would expect from a functional language, there is considerable support for list creation and processing, as well as structured concepts built on lists.

Erlang has excellent support for concurrent processes and for message passing between processes. Spawning a new process is a relatively inexpensive operation in Erlang. There is a built-in process supervisor behaviour model, where the supervising process gets notified of the termination of the subordinate and can act to restart it.

WE ADOPT ERLANG

Once we started looking at Erlang, we realized it was very well-suited for our front-end framework. Reliability of front-ends is a key concern, and Erlang is designed from the ground-up with reliability in mind. It supports the soft real-time that we need and is available on a variety of platforms, including linux that we were specifically targeting. High level data structures, such as the Erlang dict offer a painless way to implement mappings between database devices and specific controls code. Easy availability of Erlang processes, and high level language support for message passing make Erlang a natural fit for handling all the various Acnet data requests at a large variety of frequencies. Standardized process behaviours, such as supervisor or generic server, have proven useful in supporting the modularity of our design. Utilizing multiple processes and the supervisor behaviour allows us to build in overall reliability even if the implementer of a particular device's driver code has made mistakes which might cause that driver code to crash. Erlang provides an interface to C/C++ that allows us to

use C++ when we are required to read/write hardware registers that aren't accessible to Erlang.

FRAMEWORK DETAILS

One can divide our Erlang framework into several relatively independent modules: Acnet Protocol Handler, Sync, Data Acquisition Core, and Device Drivers.

Acnet Protocol Handler

We have a library of routines for connecting with the acnetd daemon which transports Acnet traffic in and out of the Erlang framework. In order to hide the details of the Acnet binary network protocols, we've developed sets of marshalling and unmarshalling routines that convert between the Acnet binary packet and Erlang records (structured lists). With a separate set of these functions for each particular Acnet message (e.g. read, set, post alarm, get plot data,...) used in conjunction with the Acnet library functions for sending or receiving messages, the rest of the Erlang code is freed from any worries about the Acnet packet protocols.

Sync

Our Erlang Sync library is our general purpose triggering system for data acquisition or other periodic tasks in Erlang. A unified interface handles read requests for periodic updates, updates on our hardware clock events, or other requests, such as on Acnet state changes. For instance, if the Erlang front-end receives a request to read data at 1 Hz, the framework registers its generic ''read data callback'' function with the Sync library and thus gets notification to run at the desired period.

Data Acquisition Core

The data acquisition tasks are the main function of the front-end. At the core here is a server process waiting on new read or set requests to arrive via Acnet. This server keeps a local mapping of the local device drivers and the database identifiers which will access those drivers. At system startup, the drivers in use are registered with the data acquisition server. Currently this is accomplished with a configuration file, one for each front-end to indicate the device types it owns, but we envision this information one day being stored in the global device database and sent to the front-end from the database at startup time.

When the data acquisition server receives a new reading request, it will generally spawn a separate process to handle that request at its desired update rate, thus keeping the central server free to receive new messages and insulating it from any buggy behaviour in the device drivers.

Alarm limit checking is also done with processing similar to a simple read, but with the obvious added limit checking added on.

Device Drivers

We refer to the code which contains the details of reading or setting any particular device as its ''device driver''. These aren't necessarily traditional linux device drivers, although that may be a part of our Erlang device driver. There is a standard Erlang programming interface that each driver conforms to. Each device driver in an Erlang front-end is registered with its local data acquisition server in order to connect to Acnet requests. This interface sends the device driver routine the details of the access request, a container of information shared throughout that device driver, and the triggering event specification.

Some controls tasks require accessing hardware which isn't reachable from Erlang code. For these tasks, we've also developed a standard method of calling from Erlang to C++, using standard Erlang-C interfaces. Our C++ programming interface lets the C++ compiler do much simple type and error checking to ensure reasonable parameters are passed to the C++ code.

PROGRAMMING IN ERLANG

Everyone on our development team is used to programming in imperative languages such as C/C++ and Java. In addition to mastering the new syntax, functional programming requires a fundamental mind-shift by the programmer. This shift is somewhat similar to changes one must make when going from a serial, procedural C program to an event-driven Java graphical user interface. The functional tools provided by Erlang also allow one to consider different solutions to a programming problem.

Two paradigms used extensively in Erlang include message passing between processes and tail recursion. Erlang's message passing is built into the language, and passing indicator atoms or message data can be done without much thought by the programmer. Since Erlang passes copies of the data, the programmer doesn't need to worry about issues like semaphore locks around shared data. Because Erlang variables are single assignment, many tasks which might be done by iteration in C are done by recursion in Erlang. The run-time system is highly optimized for tail recursion, where a function calls itself recursively and the last thing the function does is make the recursive call.

Many features of the Erlang language and run-time system only become useful as one gains some expertise with the language, for instance, the generic server (gen_server) behaviour. A novice Erlang programmer will probably spend some time writing some simple message receive loop functions. As the functionality of that loop expands, the utility of the predefined gen_server package becomes more apparent.

Functional languages by their nature encourage unit testing. Since pure functions don't have side effects, effective test routines can prove that your code works. The Erlang development system provides tools that explicitly tell you which lines of you code are tested by your unit testing functions. These coverage maps obviously help demonstrate well-tested code. The interactive Erlang shell also speeds testing because you can construct test vectors and interactively call your functions.

With reasonably modern 1-2 GHz processors running linux, Erlang's performance has been sufficient for us The Fermilab linac runs at 15Hz, so 15Hz response has been our baseline for testing Erlang system response. Using Erlang and its command-line shell, it is easy to create stressful load tests of the system, for example, setting up requests for hundreds of device readings at once. Erlang provides run-time profiling tools that help point out performance bottlenecks and show how a programming or data structure choice can influence the run-time performance.

We have not yet made use of the "hot-swap" feature that lets one upgrade a software module of a running Erlang system, but we look forward to taking advantage of this in the future.

SUMMARY

We have developed a new front-end data acquisition and controls framework in Erlang. Adapting to the functional programming paradigm has been well-worth the effort and some early deployments of Erlang front ends include control commercial motor controllers over UDP and monitoring the status of the near detector of Fermilab's Nova experiment. A few other example device drivers have also been written. We've found Erlang very suitable for this type of application and are looking at implementing other control system infrastructure in Erlang.

REFERENCES

[1] The Erlang Programming Language website at www.erlang.org.

THE FERMI@Elettra DISTRIBUTED REAL-TIME FRAMEWORK *

L. Pivetta, G. Gaio, R. Passuello, G. Scalamera, Sincrotrone Trieste, Trieste, Italy

Abstract

FERMI@Elettra is a Free Electron Laser (FEL) based on a 1.5 GeV linac. The pulsed operation of the accelerator and the necessity to characterize and control each electron bunch requires synchronous acquisition of the beam diagnostics together with the ability to drive actuators in real-time at the linac repetition rate. The Adeos/Xenomai real-time extensions have been adopted in order to add real-time capabilities to the Linux based control system computers running the Tango software. A software communication protocol based on Gigabit Ethernet and known as Network Reflective Memory (NRM) has been developed to implement a shared memory across the whole control system, allowing computers to communicate in real-time. The NRM architecture, the real-time performance and the integration in the control system are described.

INTRODUCTION

FERMI@Elettra is a light source based on a linear accelerator designed to operate two FEL undulator chains covering the wavelength range from about 100 nm to 4 nm. Designed to work at 50 Hz repetition rate, FERMI@Elettra requires the control system to be able to track and tag each shot with a unique timestamp called bunch number. Any relevant data must be tagged with the same timestamp eventually allowing to correlate machine parameters with the electron and photon beams characteristics. Moreover, a number of control loops are required to stabilize the main parameters of the beams. The control system must guarantee the mechanism, the performance and the reliability necessary to acquire all the relevant sensors, perform the feedback calculations and drive the actuators in real-time.

LINUX AND REAL-TIME

Linux, as a general purpose operating system, historically belongs to the server application field and typically privileges throughput over responsiveness. Thus, the *vanilla* linux kernel is not suitable to predict precisely the execution of a task, neither the scheduling time nor the real duration. Three different approaches have been tested to improve the real-time behaviour.

* The work was supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3

Performance Tuning

A number of methods are availabile in the plain Operating System (OS) to tune the behavior and obtain a better timing of the tasks:

- Increase the scheduling priority via the `nice` command:
  `nice -20 ./executable`.
- Change scheduling policy programmatically:
```
sched.sched_priority = \
        sched_get_priority_max(SCHED_RR)-1;
sched_set_scheduler(0, SCHED_FIFO, &sched);.
```
- In multicore processors, reserve a core to a task via the `taskset` command:
  `taskset -c1 ./executable`.
- Avoid memory pages of the process to be swapped via the `mlockall` system call.

A quick and dirty solution could be rewriting some Interrupt Service Routine (ISR) to include the desired code. The ISR cannot be scheduled and preempts any other task, except other ISRs, assuring the real-time execution. Good candidate ISRs to be modified could be those related to:

- A *data availabile* interrupt request (IRQ): when data are availabile an interrupt is raised and the critical task code could run in the end of the ISR code.
- A timer IRQ: the critical code runs on a timer period.

It is even possible to reserve a core to just one IRQ.

This approach has nevertheless an important drawback: code running in the ISR is required to be short and fast and not all applications could fit.

Preemptible Kernel

A better solution relies on the adoption of a fully preemptible kernel based on the PREEMPT_RT patch [1] and availabile on the mainline since release 2.6.23. This patch reduces the kernel code protected by *big locks*, implements the priority inheritance mechanism for in-kernel spinlocks and semaphores, adds the possibility to perform deferred operations and enables the support for High Resolution Timers (HRT). This approach leads to a much more responsive kernel/system altough it cannot guarantee an unfailing respect of the deadlines: the system is still not real-time.

Real-time Subsystem

A real-time subsystem environment cooperating with the GNU/Linux kernel provides a pervasive hard real-time support to both kernel-space and user-space applications and allows for the best performances.

For FERMI@Elettra the Adeos/Xenomai [2] real-time subsystem has been adopted. It supports several processor architectures on embedded systems, such as ARM, Blackfin, NiosII, PowerPC, x86 and, of course, runs also on standard workstations and servers. To port and optimize the Adeos/Xenomai subsystem to the embedded platform selected for the FERMI@Elettra control system [3] some work has been outsourced [4].

A complete set of Application Programming Interfaces (API) to develope real-time applications in kernel/user space, such as Inter-Process Communication (IPC), synchronization, mailboxes, etc. are availabile.

To exploit all the performance of the real-time subsystem some care must be taken: standard GNU/Linux system calls should be avoided in the real-time application and the device drivers must be eventually patched in order to use the API availabile in the real-time domain.

REAL-TIME PERFORMANCE

A laboratory testbench, shown in Fig. 1, has been set up to measure the performance of a VME based system consisting of an Emerson MVME7100 CPU and a couple of digital I/O boards.

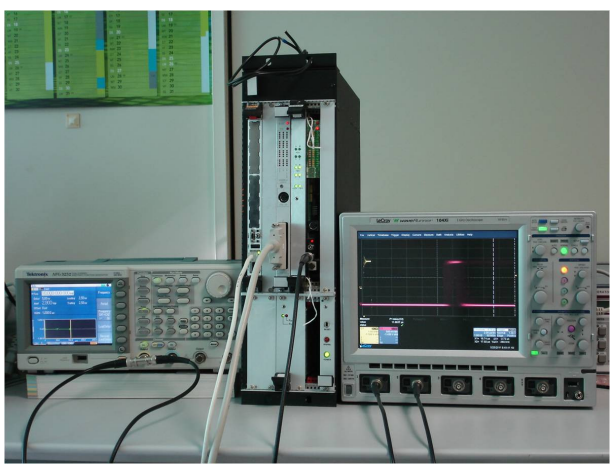


Figure 1: Real-time measurement testbench.

A pulse, referred below as *input pulse*, produced by a signal generator is acquired by a digital input board which rises an interrupt on the VME bus. The IRQ is then managed by the ISR and the digital line of an output board is driven producing an output pulse. The system has been loaded with tasks involving a large volume of interrupts: multiple *ping flood*, processes generating network activity and high-speed serial I/O traffic. Two scenarios have been analyzed.

Running the Code in the ISR

Some measurements have been carried out to characterize the performance of the described setup with the application code running inside the interrupt service routine of both GNU/Linux kernel domain and Adeos/Xenomai domain. A digital oscilloscope, triggered on the input pulse, measures the latency of the output pulse.

Table 1 and Table 2 show the results of the test for 50 Hz and 10 KHz input pulse repetiton rate respectively.

Table 1: Latency at 50 Hz Repetition Rate

	Min	Max	Mean	Std
Linux ISR	10μs	33μs	15μs	1.5μs
Xenomai ISR	11μs	23μs	16μs	1.4μs

Table 2: Latency at 10 KHz Repetition Rate

	Min	Max	Mean	Std
Linux ISR	10μs	29μs	11μs	0.8μs
Xenomai ISR	11μs	14μs	12μs	0.2μs

Measurements show that the worst case latency decreases when running at higher repetition rates, as the probability of the ISR code generating a *cache miss* decreases.

As already noticed, anyway, inserting the application code into the ISR is not good practice. The ISR must always be as fast and light as possbile.

Running the Code in an OS Task

Either the GNU/Linux kernel or the Adeos/Xenomai real-time scheduler have complete synchronization primitives that allow to execute a specific task triggered by an asynchronous event. In both cases the ISR code, servicing the IRQ, releases a semaphore that unlocks the execution of a suspended task.

A GNU/Linux task running in user space is blocked on a `ioctl` waiting on a semaphore. A GNU/Linux ISR, triggered by the IRQ generated by the digital input board, unlocks the semaphore. The GNU/Linux task generates a pulse driving a digital output line. An oscilloscope acquires both the input pulse and the output pulse. Table 3 shows the resulting measurements for a repetition rate of 10 Hz.

Table 3: GNU/Linux Task Latency Measurements

	Min	Max	Mean	Std
No load	15μs	48μs	23μs	2.5μs
Heavy load	196μs	37180μs	175μs	1374μs

As expected, the system shows outstanding performances without machine load. No guarantee,

even at 10 Hz repetiton rate, to meet the deadline when the system is heavily loaded.

In the Adeos/Xenomai case the IRQ is serviced by a Xenomai ISR. A Xenomai task is pending on a `ioctl` waiting on a Xenomai semaphore. The ISR releases the semaphore unlocking the task that drives the digital output line. The results are shown in Table 4.

Table 4: Adeos/Xenomai Task Latency Measurements

Rep. rate	Min	Max	Mean	Std
50Hz	17μs	69μs	42μs	5.7μs
10KHz	16μs	49μs	21μs	3.2μs

The real-time behaviour is guaranteed and is not affected by the laod on the system; the performance suffers a slight degradation with respect to the code running into the ISR but with a big advantage: the task runs in user space.

The Fig. 2 summarizes the measured overall latencies of the testbench. The GNU/Linux heavy load measurements are not shown.

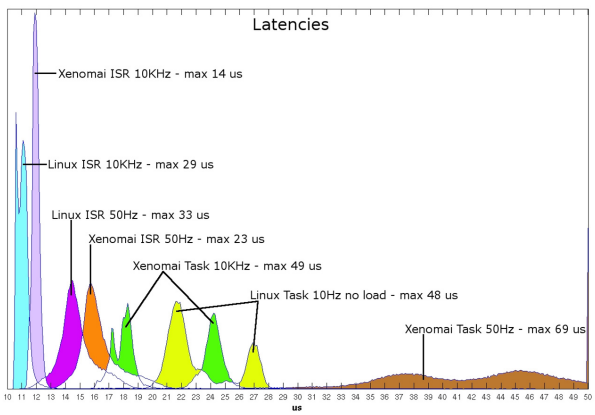


Figure 2: Overall system latency distributions.

HARDWARE

Two hardware platforms are currently supported and could be integrated into the real-time framework. VME crates, designed in accordance to the VME64x extensions, host the front-end computers; on this bus, double-edged source-synchronous transfer (2eSST) compliant boards can reach transfer rates up to 320 MB/s [5].

Emerson MVME7100 PowerPC single board computers (SBC) featuring 1.3 GHz CPU clock, 2 GB ECC RAM, 8 GB soldered flash disk, 4 Gb/s Ethernet interfaces, 5 RS232 serial lines, 2 user configurable hardware timers are used as main processors on the VME.

Intel-based processing servers have been adopted for the acquisition of digital cameras (CCD) based on Gigabit Ethernet, because of the closed-source binary-only proprietary support libraries. Two Xeon QuadCore 3.0 GHz processors, 4 GB of DDR3 RAM and up to six Gb/s Ethernet interfaces allow an effective acquisition and processing of up to three CCDs on each Gigabit link.

NETWORK REFLECTIVE MEMORY

A real-time framework tightly integrated in the control system has been implemented. Every control system cabinet is reached both by a standard control/supervisory network and by an Ethernet network dedicated to real-time. The standard Ethernet device driver has been customised in order to be able to transmit and receive raw Ethernet packets from the real-time domain. All the machines involved in real-time activities connect to this network using an additional Ethernet port.

To share the real-time data in a simple end effective way, a software infrastructure called Network Reflective Memory (NRM) has been developed. It consists of dedicated real-time device drivers and threads together with a dedicated API, availabile in both kernel space and user space, which implement the mechanisms of data-transparent memory shared among computers. Currently the device drivers of two families of Ethernet chipsets have been modified to support the NRM: the four channel Gianfar Ethernet Controller used on the MVME7100 and the Intel PRO/1000 used on the 1U rack-mount CCD acquisition servers [6].

The system topology is star shaped. At the centre a master system is in charge of collecting the packets coming from each slave and broadcasts them to all the stations. In order to guarantee the consistency of the shared memory the master broadcast is used as the trigger: upon reception, each slave syncronizes its local copy of the NRM with the data belonging to the master packet and at the same time starts sending its specific data to the master.

The NRM packets are handled in kernel space; the modified Ethernet device driver allows raw access to the interrupt handler and the transmission routines. Two First In Firts Out (FIFO) queues serve the write operations to the NRM coming from kernel space and user space respectively.

The raw packet is forged to comply to the link layer of the TCP/IP model; the header contains the MAC address of the sender and the destination, the timestamp, the sequence number and the CRC followed by the data. The data starts with a 4 bytes header containing the data type, the segment size, and the shared memory offset; the minimum payload size is therefore 4+4=8 bytes. To safeguard the availabile bandwidth shared among all stations, currently 1 Gb/s, the maximum packet size is fixed to 256 bytes for the slaves and could be increased up to the jumbo packet size for the master. Also, the address space of the NRM is limited to 1 MB.

Repetition rates up to 10 KHz are sustainable when the master and the slaves are all connected to the same network switch; otherwise, when the packets have to cross two switches the maximum repetition rate lowers to roughly 5 KHz.

An approximate calculation of the sustainable data traffic could be done considering the size of the master data packet and the usable repetiton rate; the maximum on a single Gbit Ethernet connection spanning at most two hops is roughly 9000 bytes × 5 KHz = 45 MB/s. Without using the jumbo packet payload and at the actual repetition rate, the NRM maximum data exchange rate through for the FERMI@Elettra control system is 1500 bytes × 3.3 KHz ≈ 5 MB/s.

The actual size of the NRM data currently in use is 12 KB at the Linac repetiton rate of 50 Hz, thus the data rate is 12 KB × 50 Hz = 600 KB/s. Moreover, with a NRM refresh rate of 3.3 KHz and a Linac repetition rate of 50 Hz the number of availabile NRM cycles between two shots, e.g. 20 ms, is 65. Considering the worst case of the data payload, 4 bytes data plus 4 bytes header, the number of NRM cycles used in the FERMI@Elettra control system is 12 KB / 1500 / 2 = 16 over 65, i.e. about 25% utilization.

USE CASES

A large number of devices are currently interfaced, i.e. acquired and/or driven, shot by shot:

- Electron/photon beam diagnostics: electron beam position monitors (BPM), photon beam position monitors (PHPBM), current monitors (CM), CCDs, bunch lenght monitors (BLM), beam arrival monitors (BAM), laser power meters, i0 monitors (photon counters), experimental station detectors.
- Power supplies: corrector power supplies, quadrupole power supplies.
- Radio frequency systems: linac low level RF.
- Machine protection systems: Cherenkov optic fibers, ionization chambers.

An example of a real-time FERMI@Elettra application is the Machine Protection System (MPS)[7][8]. An Equipment Controller [9] is in charge of monitoring the radiation levels along the undulator chain to avoid damaging the permanent magnets. For this purpose a number of Cherenkov optic fibers have been placed into appropriate grooves along the undulator magnets.

The waveform signals coming from the Cherenkov fibers front-ends are synchronously acquired shot by shot by a Caen V1720 VME digitizer at 250 MHz sampling rate.

A signal processing algorithm, running in real-time, analyzes the waveforms and, when over a predefined threshold, drives a digital output line requesting the switching off ot the electron beam to the MPS PLC.

Moreover, in order to monitor the operation of the software routines inside the VME system, a keep-alive signal with the repetition frequency of 50 Hz is transmitted from the VME to the PLC.

The same VME system is also in charge of the acquisition of the current monitors in real-time through a dedicated Ethernet network. Also in this case the algorithms to calculate the charge loss run in real-time on a shot by shot basis and the results are used to drive the inputs of the MPS PLC.

CONCLUSIONS

The Adeos/Xenomai realtime subsystem has been adopted to achieve hard real-time performances on GNU/Linux based systems running on PowerPC and x86 processor architectures. The measurements show that even loaded systems behave very well with the advantage of the programming API availabile in user space. The NRM has beed designed to share small amounts of data in real-time, i.e. with a known and reproducible latency, in a transparent and cheap way, leveraging the standard Gigabit Ethernet hardware. The hardware, the network topology, the software implementation of the data transmission make the NRM a good candidate for real-time distributed applications with repetition rates in the range of some hundreds of Hz and data size of tens of KB.

REFERENCES

[1] I. Molnar, `http://www.kernel.org/pub/linux/kernel/projects/rt`.

[2] `http://www.xenomai.org`.

[3] M.Lonza et al., "The control system of the FERMI@Elettra Free Electron Laser", ICALEPCS 2009, Kobe, Japan

[4] Denx Software Engineering, `http://www.denx.de`.

[5] `http://www.vita.com`.

[6] G.Gaio et al., "The FERMI@Elettra CCD image acquisition system", PCAPAC 2010, Saskatoon, Saskatchewan, Canada.

[7] F.Giacuzzo et al., "Equipment and Machine Protection Systems for the FERMI@Elettra FEL facility", these proceedings.

[8] L.Frölich et al., "Instrumentation for Machine Protection at FERMI@Elettra", DIPAC'11, Hamburg, Germany

[9] M.Lonza et al., "Status report of the FERMI@Elettra control system", these proceedings.

EMBEDDED LINUX ON FPGA INSTRUMENTS FOR CONTROL INTERFACE AND REMOTE MANAGEMENT*

B.K. Huang, Dept. of Physics, Durham University, Durham and CCFE, Abingdon, Oxfordshire, UK
G. Naylor, G. Cunningham, CCFE, Abingdon, Oxfordshire, UK
O. Goudard, ESRF, Grenoble, France
J. Harrison, Merton College, Oxford, UK
R.M. Myers, R.M. Sharples, Dept. of Physics, Durham University, Durham, UK
R.G.L. Vann, York Plasma Institute, Dept. of Physics, University of York, Heslington, York, UK

Abstract

FPGAs are now large enough that they can easily accommodate an embedded 32-bit processor which can be used to significantly extend the capability of an FPGA system. Running embedded Linux gives the user many more options for interfacing to their FPGA-based instrument, and in some cases this enables removal of the middle-person PC. It is possible to manage the instrument directly by ModBus (130 Hz poll rate and 6.6ms latency) and StarkStream (42 MB/s transfer and 83 MB/s raw throughput), and use these protocols as a low level layer to communicate with EPICS or TANGO. Additionally it is possible to use the embedded processor to facilitate remote updating of firmware which, in combination with a watchdog and network booting ensures that full remote management over Ethernet is possible.

INTRODUCTION

Embedded Linux is a very powerful tool on FPGA instruments. The reason for this is that the embedded processor is directly coupled to the FPGA logic thus giving full control using Linux. By using high level tools it is possible to create very complex systems that do not require VHDL or verilog code to be written. With a soft processor all of the logic is embedded in the FPGA, thus abstracting the design from the hardware and making it more portable helping to prevent system obsolescence.

This paper demonstrates the versatility of embedded Linux combined with two protocols: i) a low latency UDP protocol called StarkStream, ii) the widely used Modbus TCP PLC protocol. Also discussed is the strategy for inclusion to complex control systems.

EMBEDDED LINUX

Many devices now incorporate embedded Linux. The main reason for this is access to a large amount of open source software supporting a large number of hardware devices. Embedded Linux has existed on FPGA devices for at least 5 years with the Xilinx Microblaze processor currently supported in the mainline Linux kernel tree and the Altera Nios II also having a Linux port. Recent developments in the last year include support for the AXI bus which has enabled much faster data rates coupled with the use of DMA enabled drivers thus making embedded Linux more powerful than it has been before.

HIGH LEVEL TOOLS

High level tools are a very significant timesaver in developing FPGA applications and likely justify their cost. They allow access to complex DSP logic blocks such as FFT, CORDICs, etc. We have found that the MATLAB Simulink environment is very good for producing FPGA products as it has a completely open environment in which it is possible to access all the pins of the FPGA and use them in any way desired. For Xilinx FPGAs the tool is called System Generator and for Altera FPGAs the system is called DSP builder. System Generator even allows you to compile a portion of logic code and compile it as a hardware component (PCORE) of the embedded processor. This extremely useful function greatly simplifies the integration with the processor - the result is that this can be done without writing any code but solely through the use of a GUI.

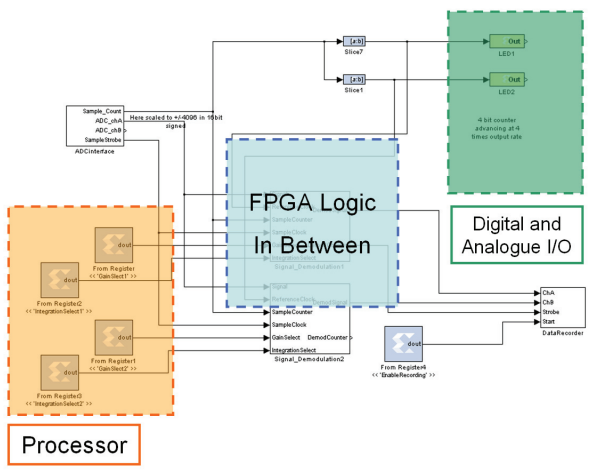


Figure 1: The high level MATLAB Simulink environment used for developing FPGA logic. In this figure the processor logic drives the FPGA logic which in turn drives the digital and analogue I/O. The digital flow can also be reversed to feed back or run completely independently of the processor.

* This work was part-funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the Contract of Association between EURATOM and CCFE.

AXI BUS

The AXI bus is an open on-chip bus. It is used by ARM, Xilinx and becoming more widely supported on Altera FPGAs. It is a very fast bus with separate read and write channels thus allowing uninterrupted data flow in both directions. It can be driven at beyond 200 MHz at 256+ bits wide which allows 51.2+ Gbps transceivers.

It is only with the AXI bus that we have been able to achieve high speed on compact systems - achieving up to 10 GB/s in our continuous data streaming application.

STARKSTREAM

StarkStream is a high speed UDP protocol. It runs on embedded Linux, and sits inside a DMA ethernet enabled driver. With interrupts on both the transmit and receive lines it is very quick to respond to sending and receiving input packets. The principle for systems using this protocol is that the FPGA board must respond as quickly as possible to any read/write requests over StarkStream.

Despite the fact that the Microblaze is only running at ∼83 MHz it is still able to achieve very high raw throughput (83 MB/s) and continuous data transfer rates (42 MB/s) that may not be expected with a soft processor. The reason for the high speed is an AXI DMA bypassing the processor for memory transfer.

Even faster speeds could be achieved by embedding the UDP packet header around the data to be transmitted. This would require FPGA logic change to add this header (no more than 64 bytes) to each 1kB up to 8kB data payload. But then the consideration has to be that the net throughput rate also depends on the receiving side, which may become the limiting factor.

REMOTE MANAGEMENT

Remote Reboot Control

Remote power control of an FPGA system is crucial for two reasons: i) Many FPGA boards are designed such that the new firmware will only be loaded once a power cycle has been performed. ii) An FPGA system with an embedded processor may have a processor which crashes (as can happen to a standard PC running an operating system). In order to be able to have full control over this we take advantage of the ATX power supply which has two critical elements, a permanent standby low current (∼100 mA) supply when the power is off and circuitry for enabling/disabling the power.

Watchdog

An Embedded Linux controlled watchdog with the poll code implemented in the heartbeat section of the kernel ensures that in the event the Linux crashes the system will automatically reboot in 5 minutes.

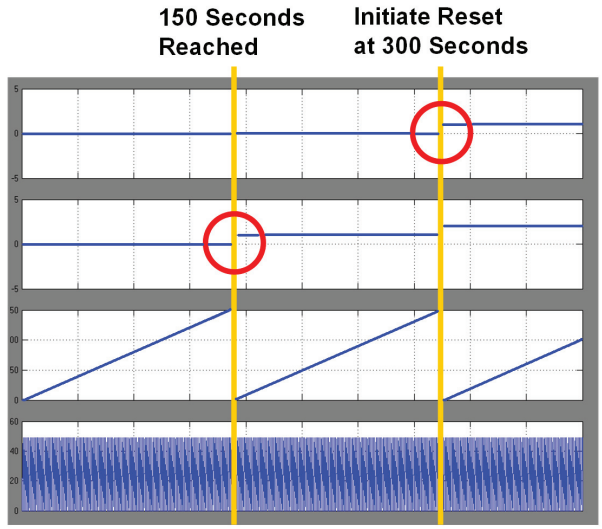


Figure 2: A Simulink simulation of the FPGA watchdog code which shows the counting ramp with a reset/reboot signal initiated after the processor has failed to poll the watchdog for 300 continuous seconds.

REMOTE FIRMWARE UPDATES

The remote firmware update uses the embedded Linux to access the storage area of the firmware (generally flash). Using embedded Linux on the FPGA provides a standard interface for updating FPGA firmware. By relying on the Linux drivers it is not necessary to develop custom drivers thus saving a lot of development time. A standard script is used to write to the specific flash locations to perform the firmware update. The system is then power cycled using the remote reboot control capability.

PLC MODBUS CONTROL

On Linux one can use the Modbus library [1] to create Modbus compatible systems. Modbus is one of the most widely supported PLC communication protocols but other protocols can similarly be adapted to the embedded Linux in a similar way.

Having embedded Linux on an FPGA systems allows it to behave as either a PLC master or slave. This gives a huge amount of flexibility to an FPGA systems adding yet another remote management interface. Since the FPGA system is running embedded Linux there is the possibility to perform other communications (HTTP/SSH/etc) in addition to PLC communications.

Latencies of 6.6ms with 0.38ms deviation (with 995 tests) have been achieved. An error rate of 3.4×10^{-6} errors/write found after approximately 35 million writes. An error is not a catastrophic failure in that the PLC master will just attempt the write again if a failure (such as a timeout) occurs. An average poll rate of 130 Hz is achieved over a period of 72 hours, resulting in a long term polling rate of at least 100 Hz.

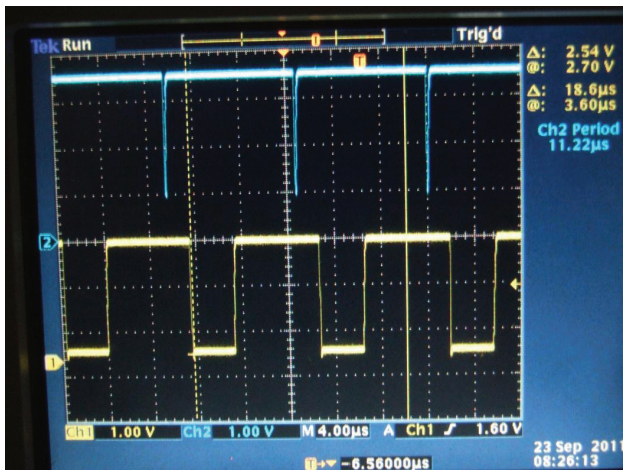


Figure 3: An oscilloscope screenshot of the Modbus FPGA System, with a configurable pulse delay and length controlled by a Schneider Modicon Quantum 140-CPU-651-60 PLC using Modbus.

DEVELOPMENT STRATEGY

EPICS and TANGO

EPICS and TANGO [2] are widely used control systems so developing an FPGA system with this capability would be useful. Some control systems have been designed to communicate over Modbus to EPICS [3, 4, 5] and TANGO [6, 7]. On an FPGA system with a soft processor the limiting factor is the processor speed since EPICS and TANGO are not lightweight. As such future development will initially involve an intermediate protocol (such as Modbus or StarkStream).

Embedded System Development

Using a generic layer between the processor and FPGA logic, it will be easier to create systems that incorporate embedded Linux with minimal effort. The goal is to have a capability to easily "drag-in" embedded Linux and synthesis the system in one step.

CONCLUSIONS

Embedded Linux has been used as a powerful tool for developing more capable FPGA systems using a standard operating system to integrate with custom FPGA logic easily using high level tools such as the MATLAB Simulink environment with the FPGA front end tools (Xilinx System Generator or Altera DSP Builder).

The AXI Bus has been used to achieve fast streaming rates (up to 10GB/s) with memory shared between the FPGA logic and the processor. Remote management has been achieved using embedded Linux to communicate over the StarkStream and Modbus protocols. In future these protocols can be incorporated with standard control systems such as EPICS and TANGO. Full remote system control has been improved by enabling firmware updates and power cycling using embedded Linux and an ATX power distribution board.

ACKNOWLEDGEMENTS

This work was funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the contract of Association between EURATOM and CCFE. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1] Stéphane Raimbault. A Modbus library for Linux, Mac OS X, FreeBSD, QNX and Win32. `http://libmodbus.org/`, 2011.

[2] A Götz, E Taurel, J L Pons, P Verdier, J M Chaize, J Meyer, F Poncet, G Heunen, E Buteau, N Leclercq, and M Ounsy. Tango a corba based control system. 2003.

[3] Ha Yu In, Myung-Hwan Chun, Ki Man Ha, Han Yeung Jin, Ha Hwang Woon, Maeng Hyo Jeong, Kang Heung Sik, Tae Kim Do, Sung-Chul Kim, I S Park, Yang Jae Seok, Yong-Sub Cho, and Tae Seol Kyung. Rfq low level RF system for the PEFP 100MeV proton linac. page 3 p, 2004.

[4] P. Gillette, C. Haquin, E. Lcorch, D. Touchard, J.F. Denis, F. Gougnaud, J.F. Gournay, Y. Lussignol, P. Mattei, P. Graehling, J. Hosselet, C. Maazouzi, and C. Olivetto. Preliminary implementation for the new SPIRAL2 project control system. *7th International Workshop on Personal Computers and Particle Accelerator Controls*.

[5] Sang Hoon Nam, Paul Bellomo, Richard Cassel, Seong-Hun Jeong, Kim Sang Hee, Sung-Chul Kim, Raymond Sverre Larsen, Minh N Nguyen, Soung Soo Park, and Jae-Hak Suh. Development of a general purpose power system control board. page 3 p, 2006.

[6] D. Fernández-Carreiras, D. B. Beltran, J. Klora, O. Matilla, R. Montano, M. Niegowski, R. Ranz, A. Rubio, and S. Rubio-Manrique. Alba the plc based protection systems. *Proceedings of ICALEPCS2009, Kobe, Japan*, 2009.

[7] D. Schmied, E. Burtin, J. M. Chaize, M. Hahn, I. Parat, M. Peru, and P. V. Verdier. A different way to survey the ESRF vacuum system. *Proceedings of ICALEPCS2009, Kobe, Japan*, 2009.

EMBEDDED LLRF CONTROLLER WITH CHANNEL ACCESS ON MicroTCA BACKPLANE INTERCONNECT

K. Furukawa*, K. Akai, A. Akiyama, T. Kobayashi, S. Michizono, T. Miura, K. Nakanishi, J. Odagiri
High Energy Accelerator Research Organization (KEK), Tsukuba, Ibaraki, 305-0801, Japan
H. Deguchi, K. Hayashi, M. Ryoshi
Mitsubishi Electric Tokki Systems, Amagasaki, Hyogo, 661-0001, Japan

Abstract

A low-level RF controller has been developed for the accelerator controls for SuperKEKB, Super-conducting RF Test facility (STF) and Compact-ERL (cERL) at KEK. The feedback mechanism will be performed on Virtex-5 FPGA with 16-bit ADCs and DACs. The card was designed in the form-factor of an advanced mezzanine card (AMC) for a MicroTCA shelf. An embedded EPICS IOC on the PowerPC core in FPGA will provide the global controls through channel access (CA) protocol on the backplane interconnect of the shelf. No other mechanisms are required for the external linkages. CA is exclusively employed in order not only to make the AMC cards to communicate with each other, but also to communicate with central controls and with an embedded IOC on a Linux-based PLC for slow controls.

INTRODUCTION

The low level radio frequency (LLRF) is one of the crucial facilities to enable the high precision and high stability of the particle accelerator beams. There are several accelerator projects planned at KEK. New projects are always demanding to achieve higher experimental results, and the LLRF control should play an important role there. After the recent development at KEK the digital LLRF controls are well accepted, and three accelerator projects began to share the developments, namely Compact ERL (cERL) as a demonstration machine for the future energy recovery linac (ERL), SuperKEKB for B-physics study, and Superconducting RF Test facility (STF) for the international linear collider (ILC). They are sharing a single LLRF controller design.

In order to achieve a higher stability for better beam quality fast digital feedback or feed-forward mechanisms are utilized on those LLRF controllers. A field programmable gate array (FPGA) is employed to carry those tasks with high precision ADCs and DACs, and it would maintain the stability of RF cavities that are covered by the controller.

On the other hand, global controls are indispensable in order to achieve overall performance of those accelerators. Sensor signals from beam instrumentation are analyzed and applied to actuators along the accelerator to improve the beam quality. Among those actuators LLRF is the fastest and the most important. Thus, the LLRF controller needs to be designed with global control capability in mind from the beginning. Globals controls are provided by EPICS (experimental physics and industrial control system) in these accelerators [1].

We designed the LLRF controller for three accelerator systems with MicroTCA platform and embedded EPICS. Even a single LLRF controller card can perform the same EPICS control functions as other I/O controllers.

* < kazuro.furukawa @ kek.jp >

LLRF CONTROLLERS AT KEK

Digital LLRF controllers were developed at KEK since 2003, and it was applied for J-PARC linac at first [2]. It was based on CompactPCI platform, and has been performing well. Based on the experiences at the J-PARC linac, LLRF controller on an advanced telecommunication computing architecture (ATCA) platform was designed for STF, because ILC chose ATCA as a possible control platform [3]. Management facilities provided by ATCA were indispensable to achieve reliable controls even with large number of accelerator components. Actually, the large board area provided by ATCA standard was adequate to accommodate a number of analog control and monitor circuits which were required by the baseline clustered klystron scheme of ILC.

The ATCA standard includes the advanced mezzanine card (AMC), and for example LLRF controllers at DESY employed AMCs on ATCA carrier card, instead of a direct ATCA card. When the LLRF controller for cERL was designed, the standard of MicroTCA was just finalized, based on AMC. The size of the standard AMC card was just enough for the small number of analog circuits at cERL. Then, MicroTCA was chosen for cERL LLRF [4]. It aimed at the future stability of 0.01 % in amplitude and 0.01 degree in phase.

It turned out that the LLRF controller at SuperKEKB shared the same requirements as one at cERL, and it was decided at SuperKEKB to employ the same design of LLRF controller [5].

Later, a new RF system configuration of distributed RF system (DRFS) was adopted at STF in the framework of S1 Global. It does not require a large card size any more compared with the clustered klystron scheme. Thus, the same LLRF card will be utilized at STF as well [6]. The choice of the controller platform at three projects above followed the synergy of the previous developments, and MicroTCA was chosen.

Computing Platforms and MicroTCA

The ATCA standard was defined in 2003 for a new computing standard for telecommunication and industry as a descendant of CompactPCI standard which was used since 1993. ATCA employed a large card area for high-performance computing, many serial interconnects on the backplane, 2.5 Gbps each, for larger bandwidths and IPMI surveillance and remote management facility for higher reliability. Even though it was designed for telecommunication and industry, the design approach of ATCA was considered adequate for ILC controls.

AMC mezzanine card was included in the ATCA standard, and AMC itself inherited many of good part of ATCA. Later in 2008, MicroTCA was standardized with direct slot-in AMC cards. As embedded controllers became smaller with the evolution of FPGA technologies, MicroTCA is considered one of the proper embedded platforms.

Both of the ATCA and MicroTCA backplanes allow fast serial communication such as Gigabit Ethernet (GbE) and PCIe. If we choose GbE as a communication media, a switch module in the box can transfer packets directly to the external networks. In MicroTCA the switch is called MCH (MicroTCA Carrier Hub), and it connects the serial backplane interconnect to an external network through a layer-2 switch.

GLOBAL CONTROLS WITH EPICS

The accelerator control architecture in KEK evolved in several steps in the past. Some time ago several control systems were standardized with a combination of several fieldbuses, VME field computers, and Unix computers. In order to consolidate the efforts on the development and maintenance some of the fieldbuses were gradually removed and many controllers were directly attached on to IP networks [7].

At the same time EPICS (Experimental Physics and Industrial Control System) control software framework was employed at several control systems at KEK. Eventually, many controllers evolved to embed the same EPICS software as on VME field computers. The EPICS component on those field computers is called IOC (Input/Output Controller), and it communicates with others using a common protocol called Channel Access (CA). We call this embedded EPICS framework as "CA Everywhere", and it enabled the both rapid development and smooth maintenance.

Several different kinds of controllers have been developed in the framework of CA Everywhere, and they greatly reduced the control efforts and improved the reliability [8].

LLRF CONTROLLER WITH EPICS CA EVERYWHERE

As described above MicroTCA can accommodate GbE and TCP/IP on the backplane interconnect directly. In addition, many field controllers recently employ the idea of CA Everywhere to embed EPICS IOC software at KEK. Thus, it was rather natural to design the new LLRF controller with embedded EPICS IOC software, and make it connect with the global control network with EPICS channel access protocol directly through the backplane as in Fig. 1.

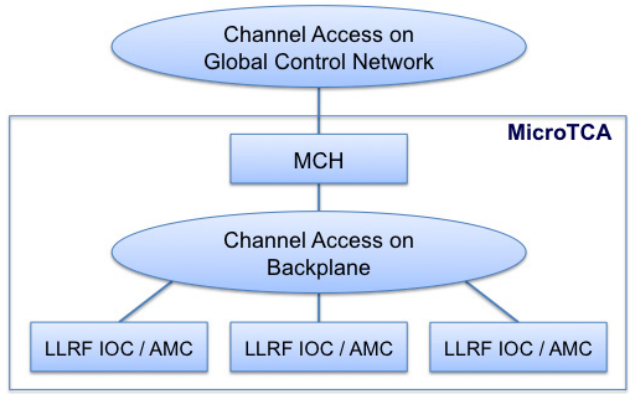


Figure 1: EPICS IOC software is embedded in controller cards and EPICS channel access protocol is used everywhere.

The PowerPC CPU in Virtex-5 FPGA on the LLRF controller runs Linux 2.6.21, and EPICS IOC 3.14.9 is embedded. The Linux kernel is booted from the on-board flash memory. On the other hand EPICS IOC software is loaded from a NFS file server, as the software can be improved frequently. Device support software to link between FPGA and IOC was developed using mmap() interface.

Hardware Features

The controller module was fabricated in a single-width full-height AMC module with two analog boards and a digital board, as shown in Fig. 2. The analog boards comprise four channels of 16-bit 130 Msps ADCs, four channels of 16-bit 500 Msps DACs. The digital board accommodates a Xilinx Virtex-5 FXT FPGA with PowerPC-440 hardcore CPU, 640MB RAM, 64MB flash memory, digital I/Os, IPMB, and GbE. IPMB (intelligent platform management bus) is used to manage the AMC in order to improve the reliability.

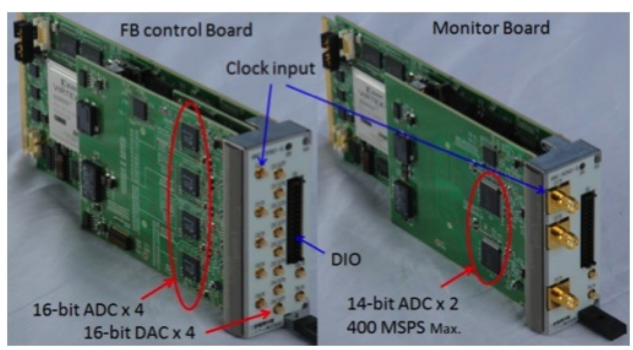


Figure 2: LLRF controller and monitor cards.

The digital part was designed based on the evaluation board of Xilinx ML507. The software for the controller was initially developed on ML507, and the main part of Linux implementation was provided by Wind River Linux 2.0.

Besides the feedback controller card, a LLRF monitor card was designed with the same digital board but with another analog board of two channels of 14-bit 400 Msps ADCs.

Other EPICS Embedded Controllers

Slow controls for cavity vacuum and so on are provided by a PLC system. It is equipped with Yokogawa's F3RP61 CPU with Linux 2.6.26, and EPICS IOC 3.14.11 is embedded on it.

The machine protection system at cERL employed a modular system which is served by a commercially available FPGA card. The card consists of Xilinx Virtex-4 with PowerPC and Linux 2.6, that embeds EPICS IOC 3.14. This card is used for other controllers as well.

The same protocol and the same software environment are shared between LLRF controllers and external servers and clients. That enables a reliable control system. The system structure is described in Fig. 3.

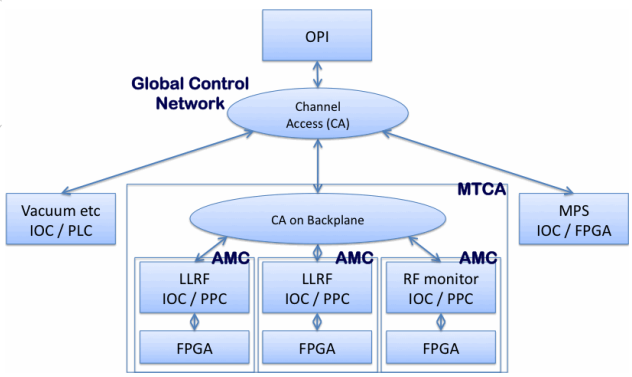


Figure 3: LLRF controller AMC cards can communicate locally on the MicroTCA backplane interconnect and globally through normal IP networks both using EPICS channel access protocol.

Enhancement Possibility

MicroTCA inherits several mechanisms from ATCA to enhance the availability, which can be utilized in order to improve the reliability of the LLRF system. Redundant power supplies, MCHs and CPUs are being evaluated as in Fig. 4. Presently, processes in LLRF system is carried on the IOCs on controllers. Additional CPU modules can be introduced into MicroTCA box to facilitate local processes among several controllers. We may run EPICS redundant IOCs on those redundant CPUs to enhance the reliability further [9].

After this controller was developed, the MicroTCA physics extension (MTCA.4) was defined. This platform may relieve the busy set of three circuit boards in the controller. We may utilize this new standard in the future.

Figure 4: Redundant CPUs, MCHs and power supplies are evaluated as a possible future extension.

The combination of fast ADCs, DACs and FPGA may serve as a general controller. As a connection to global EPICS controls was also provided, this controller can be applied for many different purposes in machine controls. Actually, the same controller card is evaluated for a specific beam monitor at SuperKEKB.

CONCLUSION

As a natural consequence of several developments at KEK, a LLRF controller in MicroTCA AMC with EPICS CA protocol on the backplane interconnect was developed. All controllers embed EPICS IOC to share the same software and protocol both locally and globally for rapid developments and smooth maintenance, including auxiliary controllers with PLCs and FPGA modules. This LLRF system is being applied to three accelerator projects of SuperKEKB, cERL and STF at KEK.

REFERENCES

[1] EPICS: <http://www.aps.anl.gov/epics/>.

[2] S. Michizono *et al.*, "Digital Feedback System for J-PARC Linac RF Source", *Proc. Linac2004*, Luebeck, Germany, 2004, THP57, p. 742.

[3] J. Carwardine *et al.*, "The ILC Global Control System", *Proc. PAC2007*, Albuquerque, USA., 2007, TUZAC01, p. 868.

[4] T. Miura *et al.*, "Low Level RF System for cERL", *Proc. IPAC2010*, Kyoto, Japan, 2010, TUPEA048, p. 1440.

[5] J. Odagiri *et al.*, "Fully Embedded EPICS-based Control of Low Level RF System For SuperKEKB", *Proc. IPAC2010*, Kyoto, Japan, 2010, WEPEB003, p. 2686.

[6] S. Michizono *et al.*, "Digital LLRF System for STF S1 Global", *Proc. IPAC2010*, Kyoto, Japan, 2010, TUPEA048, p. 1437.

[7] K. Furukawa, "Modern Accelerator Control Systems", *Proc. PAC2007*, Albuquerque, USA., 2007, TUZAC02, p. 873.

[8] K. Furukawa *et al.*, "Control System Achievement at KEKB and Upgrade Design for SuperKEKB", *in these proceedings*.

[9] A. Kazakov *et al.*, "Using EPICS Redundant IOC in Unix Environment", *Proc. ICALEPCS2007*, Knoxville, USA., 2007, TPPA31, p. 158.

MARTE FRAMEWORK: A MIDDLEWARE FOR REAL-TIME APPLICATIONS DEVELOPMENT

A. Neto, D. Alves, B.B. Carvalho, P.J. Carvalho, H. Fernandes, D.F. Valcárcel,
Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear - Laboratório Associado,
Instituto Superior Técnico, Universidade Técnica de Lisboa,
1049-001 Lisboa, Portugal
F. Sartori, Fusion for Energy, Barcelona, Spain
A. Barbalace, G. Manduchi, Euratom-ENEA Association, Consorzio RFX,
35127 Padova, Italy
L. Boncagni, Associazione EURATOM-ENEA sulla Fusione, C.R. ENEA Frascati,
I-00044 Frascati-Rome, Italy
G. De Tommasi, Associazione EURATOM-ENEA-CREATE,
Via Claudio 21, 80125, Napoli, Italy
P. McCullen, A. Stephen, EURATOM-CCFE Fusion Association, Culham Science Centre,
Abingdon OX14 3DB, United Kingdom
R. Vitelli, Università di Roma, Tor Vergata, Via del Politecnico 1-00133, Roma, Italy
L. Zabeo, ITER Organisation, Cadarache, France
JET EFDA Contributors*, Culham Science Centre, OX14 3DB, Abingdon, UK

Abstract

The Multi-threaded Application Real-Time executor (MARTe) is a C++ framework that provides a development environment for the design and deployment of real-time applications, e.g. control systems. The kernel of MARTe comprises a set of data-driven independent blocks, connected using a shared bus. This modular design enforces a clear boundary between algorithms, hardware interaction and system configuration.

The architecture, being multi-platform, facilitates the test and commissioning of new systems, enabling the execution of plant models in offline environments and with the hardware-in-the-loop, whilst also providing a set of non-intrusive introspection and logging facilities. Furthermore, applications can be developed in non real-time environments and deployed in a real-time operating system, using exactly the same code and configuration data.

The framework is already being used in several fusion experiments, with control cycles ranging from 50 microseconds to 10 milliseconds exhibiting jitters of less than 2%, using VxWorks®, RTAI or Linux. Codes can also be developed and executed in Microsoft Windows® and Solaris®.

This paper discusses the main design concepts of MARTe, in particular the architectural choices which enabled the combination of real-time accuracy, performance and robustness with complex and modular data driven applications.

* See the Appendix of F. Romanelli et al., Proceedings of the 23rd IAEA Fusion Energy Conference 2010, Daejeon, Korea

INTRODUCTION

MARTe [1] is a framework tailored at the design and development of real-time control systems. The kernel of MARTe uses a C++ multi-platform library named BaseLib2 [2]. The main ideas behind the original design of the framework were the modularity and portability of its applications. In particular, a strong effort was made in order to allow a robust simulation environment that allows both the models to be simulated with the hardware in the loop and the control algorithms to be validated offline.

This is achieved by providing a clear development boundary between the algorithms, hardware and system configuration. Being multi-platform it also allows to debug and develop in non-real-time environments, where better developing tools are usually available. Currently the framework runs in Linux, Linux with RTAI [3], VxWorks® for PowerPC®, Solaris® and Microsoft® Windows™.

The framework components are configured using a common language, designed to be as simple as possible, but complete enough to provide a clear way of describing the problem. The structure is similar to XML, where the syntax rules are validated by a BaseLib2 parser, whereas the actual validity of the arguments is performed by the component (i.e. no validation schema is available). An example of a configuration is shown in Listing 1.

The configuration file is translated into a database of named objects that can be browsed using the object addresses in the database. By parsing the configuration file, the framework automatically creates and configures instances of all the declared objects. A messaging mechanism uses the database to provide a standard interface for communication between objects. This is the preferred

```
+HttpServer = {
   Class = HttpService
   Port  = 8084
}
+MARTe = {
   Class = MARTeContainer
   +RTThread1 = {
      Class = RealTimeThread
      +SurfaceTemperature_WOPL_14 = {
         Class = SurfaceTemperature1DCalculation
         SpecificHeat = 1.925000e+03
         ThermalConductivity = 1.900000e+02
         DeltaT = 0.01
         Nslices = 40
         InputSignalNames = {
           0 = {Q_WOPL_14}
         }
         OutputSignalNames = {
           0 = {SurfaceTemperature_WOPL_14}
         }
      }
      +DivertorThermalCalculations = {
         ...
         WallsPowerPartitionClass = {
           Type = WallsPowerPartitionClass
           PartitionTable = {
             0 = {
               0 = {0.3  0.3  0.3  0.3}
               1 = {0.7  0.7  0.7  0.7}
             }
             ..
           }
         ...
      }
   }
...
```

Listing 1: A small fraction of a configuration file from the real-system responsible for the JET plasma wall load protection system (WALLS).

mechanism to interface MARTe objects with other systems and protocols.

MAIN COMPONENTS

A MARTe application is designed by configuring and connecting a series of blocks named Generic Application Modules (GAM). These modules contain an entry point to receive data driven configuration and a set of data-driven optional input and output channels to interface with other GAMs. Each of these channels has a unique name and is connected to a generic memory data pipeline, named Dynamic Data Buffer (DDB). Before starting the execution of the application, MARTe guarantees the coherency of the DDB by checking that all GAMs have the requested inputs being produced by another GAM. This scheme enables to design interchangeable and generic modules, that can be used in different projects without knowing any details or imposing any restrictions in the data producer. Moreover, this is also the key design concept which enables to replace a part of the system by a set of simulation GAMs, without changing the other modules, a very important feature when testing and designing a new control system, before introducing the hardware in the loop, and later in the commissioning of new hardware if good models of the plant are available. An example is shown in Fig. 1. A large set of generic GAMs is available to be used in any MARTe application (e.g. PID, waveform generation, live data view, data collection, data statistics).

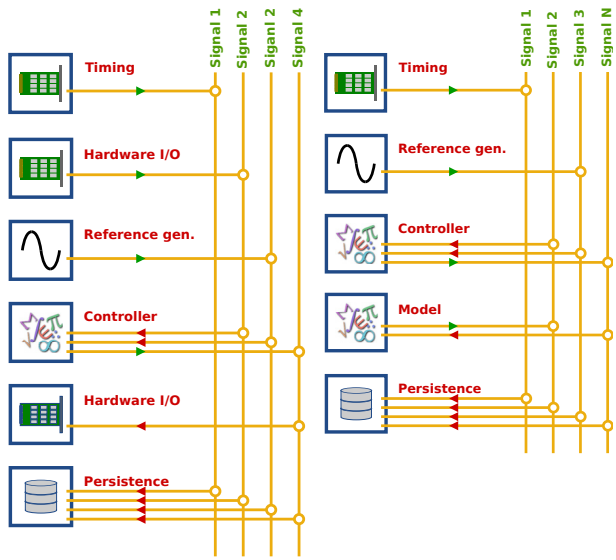


Figure 1: Example of a set of GAMs connected to the DDB. A timing and an hardware GAM provide the I/O interface to the outside world, whereas a generic waveform GAM inputs the reference for a PID controller. Finally, the output is sent to a DAC and the data is stored for analysis by a collection GAM. The picture in the right shows how the same system can be developed using a model of the plant. It should be noticed that the reference generation and the controller GAM are not aware of the changes in the data providers and data consumers.

A special GAM, named IOGAM, enables the connection of any hardware to the DDB, as long as a MARTe high level driver is developed to provide the connection between the IOGAM and the hardware interface (usually through an operating system low level driver). GAMs are sequentially executed, at a given frequency, by a real-time thread. Several real-time threads can be connected, both synchronously and asynchronously, and executed in parallel.

A real-time thread cycle is triggered by an entity named external time triggering service (ETTS). The ETTS is connected to a time provider and checks that the current time is a multiple of the configured MARTe cycle time. When this condition holds true a new cycle is signalled. Two types of ETTS are available: one based in hardware interrupts and another based in the polling of a resource (e.g. shared memory entry). The synchronisation scheme will issue an alarm if a timeout occurs, either due to a slow execution of the control cycle (e.g. a GAM is consuming too much time and the cycle finishes after its period has already elapsed) or if the jitter in the timing source is very high. MARTe

provides some ready to be used synchronisation schemes based in CPU timers and in network inputs.

Taking advantage of the new multi-core processors, the framework enables all of its tasks to be allocated to a specific subset of cores. In Linux, using the special *isolcpus* kernel parameter, it enables to have an isolated real-time execution environment for the real-time threads with jitters in the order of the microseconds [4].

INTERFACING WITH MARTE

All MARTe objects are setup using the configuration mechanism described above. The framework makes no assumptions on how these files are produced and does not impose any communication protocol on how these should be transmitted. A standard C++ interface, with all the infrastructure from the MARTe side implemented, is available for the development of new communication mechanisms. Currently, for systems outside JET, the only ready to be used configuration mechanism is based in the HTTP protocol, although the preferred way of interfacing with MARTe components is to use the message mechanism introduced before. In order to integrate the framework in different environments, the required protocols are translated into messages and broadcast to the destination objects. An example of such integration can be found in [5].

The framework also provides its own HTTP server, capable of browsing and introspecting any of the installed objects. The server was designed to minimise any impact with the real-time activities, by carefully executing all of its activities in low-priority tasks and, in the case of multi-core environments, in cores not allocated for the real-time threading. This scheme enables GAMs to publish run-time execution information about the internal state of algorithms and data.

MARTE SYSTEMS

As shown in Table 1, a large number of control systems is already using MARTe to solve different control problems [6]. These systems are key to the operation of large experiments, and some have an active role providing a first line of defence for the protection of the machine itself. The framework is installed in different experiments, with different configuration and data retrieving protocols.

The most frequent operating systems are Linux, running in Intel® and AMD® processors with isolated cores, and VxWorks® running in PowerPC®. The frequency of execution spans from 100 Hz to 20 kHz. Some modules, like the waveform generation, data collection and data statistics are shared by all projects.

The COMPASS control system [8], depicted in Fig. 2, was the first to exploit the capability of running multiple real-time threads at different frequencies. The fastest thread executes at 20 kHz and starts by reading the data from the analogue input channels, followed by a drift compensation (due to the use of integrators). Data is then filtered and sent to the second, slower, thread running at 2 kHz. In parallel, the fastest thread controls the horizontal and vertical magnetic fields, while the slower thread is responsible for the control of the plasma current and position. Since the hardware is the same used by the JET vertical stabilisation, the only GAMs that had to be developed concerned the plasma control.

Table 1: Systems using MARTe

Name	Cycle time	O.S.
JET Vertical Stabilisation [1]	50 µs	Linux-RTAI
JET Error Field Correction Coils [7]	200 µs	VxWorks®
COMPASS [8] Shape Controller	500 µs	Linux
COMPASS [8] Vertical Stabilisation	50 µs	Linux
ISTTOK [9] Tomography	100 µs	Linux
FTU [10] Plasma Control	500 µs	Linux-RTAI
RFX [11] MHD Control	125 µs	Linux
JET Real-time Protection Sequencer [12]	2 ms	VxWorks®
JET Vessel Thermal Map [13]	10 ms	Linux
JET Plasma Wall Load System	10 ms	Linux

CONCLUSIONS

MARTe is C++ real-time framework designed for the development and deployment of control-systems. It is already being used in several experiments to solve different problems that require different operational frequencies and hardware interfaces. Currently it is only being used in magnetic confinement nuclear fusion experiments, but there is not reason why it cannot be deployed in any context where a real-time control system is required.

ACKNOWLEDGEMENTS

This work was supported by the European Communities under the contract of Association between EURATOM/IST and was carried out within the framework of the European

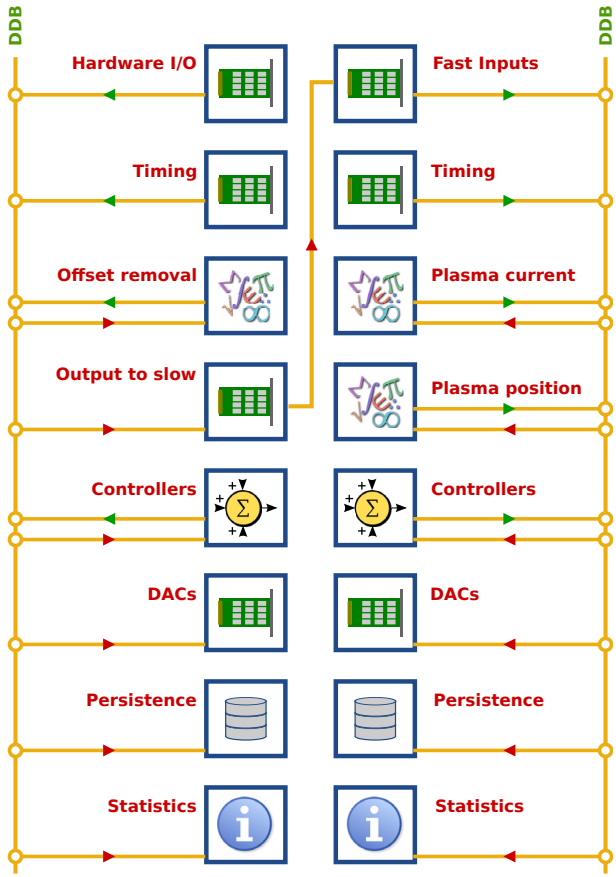


Figure 2: The COMPASS plasma control system was the first system to exploit the multi real-time threading capabilities of MARTe. A fast thread (20 kHz) provides the ADC data to a slower thread (2 kHz).

Fusion Development Agreement. See the Appendix of F. Romanelli et al., Proceedings of the 23rd IAEA Fusion Energy Conference 2010, Daejeon, Korea. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1] A. Neto et al., "MARTe: A Multiplatform Real-Time Framework", IEEE Transactions on Nuclear Science, Vol. 57, pp. 479-486, 2010.

[2] G. De Tommasi et al., "A flexible software for real-time control in nuclear fusion experiments", Control Engineering Practice, Vol. 14, pp. 1387-1393, 2006.

[3] A. Neto et al., "Linux real-time framework for fusion devices", Fusion Engineering and Design, Vol. 84, pp. 1408-1411, 2009.

[4] D.F. Valcarcel et al., "The COMPASS Tokamak Plasma Control Software Performance", IEEE Transactions on Nuclear Science, Vol. 58, pp. 1490-1496, 2011.

[5] D.F. Valcarcel et al., "EPICS as a MARTe Configuration Environment", IEEE Transactions on Nuclear Science, Vol. 58, pp. 1472-1476, 2011.

[6] A. Neto et al., "A Survey of Recent MARTe Based Systems", IEEE Transactions on Nuclear Science, Vol. 58, pp. 1482-1489, 2011.

[7] D. Alves et al., "The new Error Field Correction Coil controller system in the Joint European Torus tokamak", accepted for publication in Fusion Engineering and Design.

[8] D.F. Valcarcel et al., "Real-time software for the COMPASS tokamak plasma control", Fusion Engineering and Design, Vol. 85, pp. 470-473, 2010.

[9] P.J. Carvalho et al.,"ISTTOK plasma control with the tomography diagnostic", Fusion Engineering and Design, Vol. 85, pp. 266-271, 2010.

[10] L. Boncagni et al., "First Steps in the FTU Migration Towards a Modular and Distributed Real-Time Control Architecture Based on MARTe", IEEE Transactions on Nuclear Science, Vol. 58, pp. 1778-1783, 2011.

[11] A. Barbalace et al., "Concepts, Design, and Development of a Multiplatform Framework for Real-Time Control in Nuclear Fusion", IEEE Transactions on Nuclear Science, Vol. 57, pp. 688-695, 2010.

[12] A. Stephen et al., "Centralised Coordinated Control To Protect The JET ITER-like Wall", ICALEPCS2011, Grenoble, France.

[13] D. Alves et al., "The Software and Hardware Architectural Design of the Vessel Thermal Map Real-Time System in JET", ICALEPCS2011, Grenoble, France.

DEVELOPMENT OF THE MACHINE PROTECTION SYSTEM FOR LCLS-I*

J. Dusatko#, M. Boyes, P. Krejcik, S. Norum[1], J. Olsen
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, U.S.A.

Abstract

Machine Protection System (MPS) requirements for the currently operating Linac Coherent Light Source (LCLS-I) demand that fault detection and mitigation occur within one machine pulse (1/120th of a second at full beam rate). The MPS must handle inputs from a variety of sources including loss monitors as well as standard state-type inputs. These sensors exist at various places across the full 2.5Km length of the accelerator and beam lines. A new MPS has been developed based on a distributed star network where custom-designed local hardware nodes handle sensor inputs and mitigation outputs for localized regions of the LCLS accelerator complex. These Link-Nodes report status information and receive action commands from a centralized processor running the MPS algorithm over a private network. The individual Link-Node is a 3u chassis with configurable hardware components that can be setup with digital and analog inputs and outputs, depending upon the sensor and actuator requirements. Features include a custom MPS digital input/output subsystem, a private Ethernet interface, an embedded processor, a custom MPS engine implemented in an FPGA and an Industry Pack (IP) bus interface, allowing COTS and custom analog/digital I/O modules to be utilized for MPS functions. These features, while capable of handling standard MPS state-type inputs and outputs, allow other systems like beam loss monitors to be completely integrated within it. To date, four different types of Link-Nodes are in use in LCLS-I. This paper describes the design, construction and implementation of the LCLS MPS with a focus on the Link-Node, which has proven to be a very useful and flexible component for the MPS.

HISTORICAL PERSPECTIVE

The SLAC National Accelerator Laboratory accelerator complex has hosted several different Machine Protection Systems throughout its varied history. The original 1960s era SLAC Linac contained two protection networks [1]. The primary one, capable of responding within 1ms, consisted of a transmitter broadcasting two tones through a network of series connected Tone Interrupt Units (TIUs) to a receiver unit, which gated beam production on/off at the injector. The TIUs used mechanical relays actuated by various system sensors, so that if a sensor tripped, the tone signal path would get interrupted, inhibiting triggers and indicating a fault alarm.

The SLAC Linear Collider (SLC) era in the 1980s ushered in a more sophisticated MPS [2] where programmable logic algorithms and more sophisticated sensors allowed automatic limiting and control of beam repetition rates without switching off the entire accelerator. This system was based on both CAMAC and VME bus platforms and used the MIL-STD-1553 serial bus standard as a fast, dedicated link between the sensor units and fault processor units. This became known as the "1553 MPS". It interfaced with the accelerator timing system, dictating beam rates to it. The 1553 MPS was largely independent and parallel to the TIU MPS.

The development of LCLS-I presented a unique opportunity and challenge to replace both of these legacy systems by combining and integrating their functionality. The new LCLS-I MPS had to connect to legacy accelerator systems as well as to completely new installations built solely for LCLS-I. The development effort also provided the opportunity to use modern technology.

THE LCLS-I MPS

SLAC is building a second facility, LCLS-II, so we now refer to the present machine as LCLS-I. The LCLS-I Free Electron Laser is a pulsed machine with a maximum repetition rate of 120Hz. LCLS-I uses the last 1/3rd of the SLAC LINAC, the existing switchyard beamline and then extends this to a newly constructed undulator hall e-beam dump, photon transfer lines and experimenter areas. An overview of the LCLS-I MPS is given in [3]. The system requirements for the LCLS MPS [4], state that the MPS will only turn off the electron beam and not any other devices. In addition to turning off the beam, this system also needed to be capable of rate limiting beam production upon certain conditions. System inputs include standard (mechanical and electronic switch state) sensors (e.g. valve position) as well as direct input from beam loss detectors. Presently the input device count is approximately 2100.

The LCLS MPS has outputs to control beam mitigation and shutoff actuators. These devices are mechanical shutters at the photoinjector laser and laser heater, as well as the gun timing trigger permit. Another mitigation device is a fast pulsed kicker magnet located just prior to the undulator magnet string.

The minimum required response time (sensor in to mitigation out) is 8.3mS. As implemented, the system actually samples, processes and acts on sensors at 2.78ms.

The architecture is that of a star network with a central MPS Link Processor connecting to individual Link-Node units. The interconnect channel is a private Gigabit Ethernet network which uses commodity Ethernet switches (Cisco Catalyst 3750). The Link Processor (LP)

*Work supported by U.S. Department of Energy under Contract Nos. DE-AC02-06CH11357 and DE-AC02-76SF00515.
[1]Now at Varian Medical Systems, Palo Alto, CA
#jedu@slac.stanford.edu

implements the MPS algorithm and is responsible for all decision making based on the states of device inputs it receives from each of the Link-Nodes (LN). The LNs provide the I/O interface between the MPS and the outside world; with a LN connecting to sensors (input devices) and mitigation units (output devices). The LP (a Motorola MVME-6100 CPU) also communicates rate limiting commands to the LCLS Timing System [5] via a point-to-point Ethernet link. It receives timing pattern and fiducial information from the timing system by way of a connected timing Event Receiver (EVR) module. The EVR (MicroResearch Finland PMC-EVR-200) generates an interrupt to the LP at the timing system 360Hz fiducial rate. The LP also connects to the regular control system Ethernet network (via a second Ethernet port) for status and control communication.

Communication between the LP and LNs over the private Gigabit Ethernet network uses the UDP protocol. The upper layers of the Ethernet protocol were not used in order to keep the overhead low. The communications sequence is as follows: 1) The LP is woken up by the 360Hz interrupt generated by the timing fiducial received by the EVR. 2) The LP sends a sync broadcast message to all LNs, asking them to return their status. 3) The LNs then send their device state status information to the LP. 4) The LP sends a permit command message telling specific mitigation devices to actuate based on the previous (one 2.78ms tick) sensor status. 5) The LP processes the new status and sends out a status clear message back to the LNs, telling them to clear any latched faults. 6) The LP, based on its algorithm, processes the current sensor states and decides if mitigation devices need to fire on the next fiducial. 7) The LP then waits for the next interrupt. Software and FPGA watchdogs ensure a fail-safe communications link. A maximum of 254 LNs can be connected to the LP. The LN IP address is set with DIP switches. Packet size is modest (< 1000 bytes) and the data payload is small enough to handle communications at 360Hz without dropped or split packets. Maximum link communication times were calculated and later empirically confirmed (< 1ms). The LNs also connect (via its embedded CPU) directly to the control system network for control, status and housekeeping.

THE LINK-NODE

The Link-Node is the central functional element of the LCLS MPS and its unique design merits further elucidation. In order to accommodate a variety of input signals distributed across the full accelerator complex, a local signal collection and processing point was required. It was desirable to develop a solution that could accommodate a variety of inputs, which led to the idea of a modular platform. Several crate-based platforms were considered including VMEbus and micro-TCA. The chassis design ultimately won out based on cost and the need for customizability. This gave a common platform that could be configured in multiple ways with low overhead. A block diagram of the Link-Node chassis is given in Fig. 1. The chassis consists of several circuit boards, and depending upon the flavour of LN (Mitigation, BLM, PIC, ByKIK) the configuration and types of these boards varies. The general functions of each of these boards will now be described

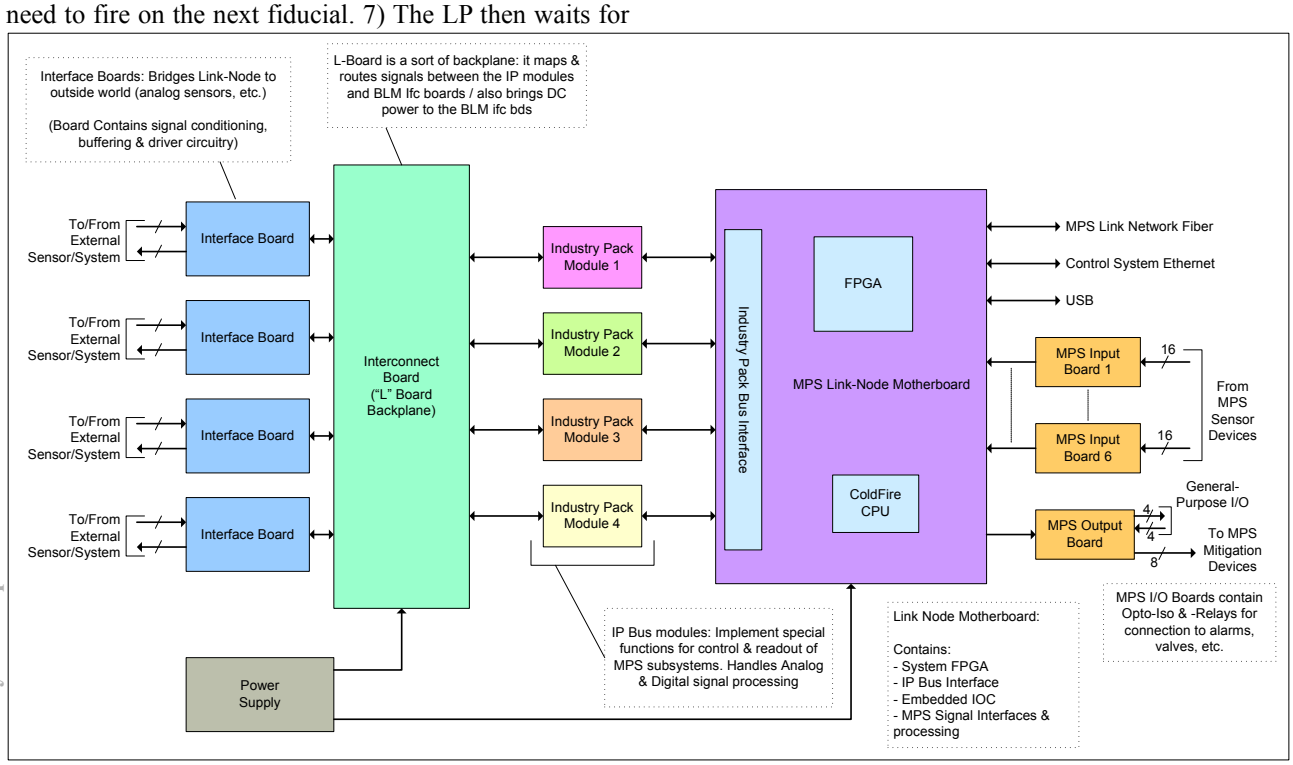


Figure 1: MPS Link-Node Block Diagram.

Motherboard

The Motherboard (MB) contains the main functions of the LN chassis. It contains a Xilinx XC4VFX20 Virtex-4 FPGA, a ColdFire embedded CPU (Arcturus Networks uC5282), the MPS digital I/O interface, a USB 1.0 interface and the Industry Pack bus interface. The USB interface permits local control and status readout for development and maintenance purposes. The FPGA implements the MPS engine, the private Ethernet interface (using its embedded MAC and Rocket I/O cores), the USB interface and the IP bus interface. The MPS sensors and mitigation devices connect to the motherboard via the MPS Digital Input and MPS Digital Output boards, respectively.

Input and Output Boards

The MPS Digital Input Board accommodates 16 optically-coupled inputs. It is capable of supplying up to 2A of fuse-protected standard +24VDC for interface logic. A total of six Input Boards can be installed into the MB, giving the L-N chassis up to 96 digital inputs.

The MPS Digital Output Board contains eight optical relay outputs for driving mitigation devices. It is also provides up to 1A of fused +24VDC for interface logic. The MB can accommodate only one Output Board. This board also contains four each, general-purpose digital inputs and outputs; available on Lemo connectors. The inputs are 50Ω terminated TTL compatible schmitt-trigger types, while the outputs are high current TTL drivers, capable of driving up to 67mA into 50Ω. These I/Os are configurable for different functions inside the MB FPGA and are used for dedicated I/O in the BLM, PIC and ByKIK L-N models.

Industry Pack Interface and Modules

A standard Industry Pack (IP) Bus interface is available on the MB that can accommodate up to four IP modules. This interface allows the use of custom and COTS IP modules to enable a signal processing, status and control path between sensor elements and the MPS. With this capability we can completely integrate a sensor readout subsystem within a LN chassis. This enables the MPS to directly connect to the beam loss and pulsed kicker magnet systems. The IP bus side of the IP modules connect to LN MB, while the I/O side of the IP modules connect to the L-board. The IP modules receive their power from the MB. The type of IP board used depends on the flavour of LN, and can be analog and/or digital I/O.

L-Board and Interface Board

The L-Board, so named for its physical shape, is essentially a backplane that provides interconnect and power between the IP modules and the interface boards. There is one L-Board per chassis. It contains mostly connectors for power, the IP and Interface boards and I/O of key IP module signals. To date, two L-boards have been designed: the BLM and PIC L-boards.

The Interface Board contains signal conditioning, receiver, driver and protection circuitry that bridges the LN to external devices. It receives its power from the L-board. The Interface Board is application-specific and to date, two Interface Boards have been designed: the BLM and PIC Interface Boards.

Motherboard FPGA

The MB FPGA implements the core functions of the LN. A block diagram of the FPGA is given in Fig. 2.

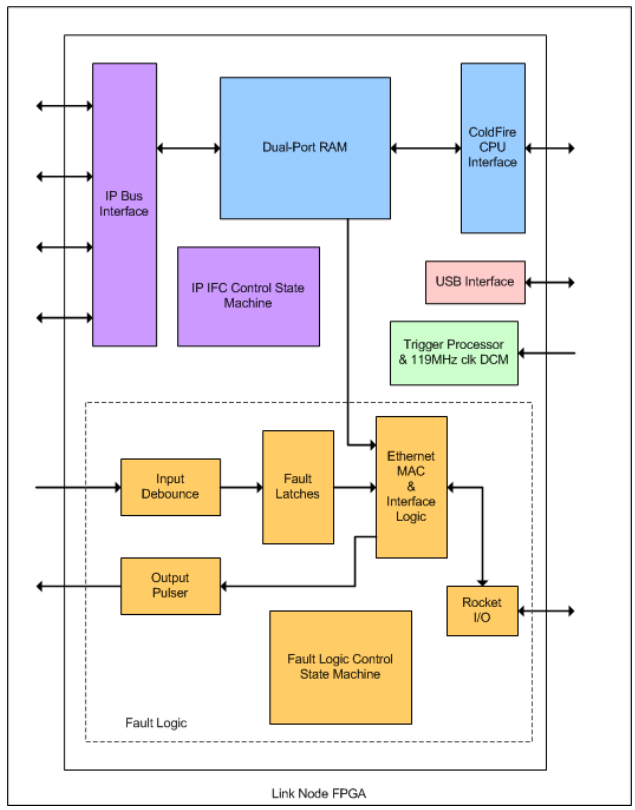


Figure 2: LN Motherboard FPGA Block Diagram.

Conceptually, the FPGA is divided into sections. The Fault Logic, inside the dotted box, implements the MPS engine. This logic contains all of the fault processing and communications functions including the Ethernet interface. The sensor inputs are passed into a debouncer block which has eight programmable debounce times ranging from 10µs to 5s; this block helps filter out noise (e.g. bubbles on waterflow switches), preventing false trips. The debouncer feeds the fault latch logic, whose output is sent to the LP. The output logic consists of a pulse generator which generates a 9ms pulse when a mitigation command is received. This output must be refreshed at each communications sequence, otherwise it will time out. This is done to avoid a latched failure mode in the case of a link failure. The fault logic is controlled by a state machine which sequences operations and communication. The fault logic also receives inputs from the IP interface dual-port RAM block. For certain applications, the IP modules can provide fault information. This fault data is passed to the Ethernet

interface so that the LP can receive it. The ColdFire interface is used to control the IP modules and other LN chassis housekeeping, control and status functions. In addition, the ColdFire has access to the fault latch data in a read-only manner.

LINK-NODE FLAVORS

This section describes each of the types of Link-Node currently being used in the system. It should be noted that in all configurations, the MPS Digital I/O boards can be installed.

Mitigation Link-Node

The Mitigation LN is the simplest configuration. It consists of just the motherboard and MPS Digital Input/Output boards. The configuration of I/O boards is done on a per-instance basis depending upon device requirements.

BLM Link-Node

The Beam Loss Monitor LN is the most complicated. It is part of the BLM readout system [6]. This LN can connect to up to eight undulator BLM detector head and front end electronics box pairs. It is a triggered device, receiving triggers from the timing system [5]. A custom eight-channel IP module, the IP-QINT-ADC (QADC), was developed at SLAC to digitize and process the detector signals. The BLM LN sets and reads back the PMT high voltage power supplies (located in the front end electronics) using COTS DAC and ADC IP modules. The LN also generates a test pulse to exercise the BLM PMT & readout chain. The BLM's high voltage and fault thresholds are setup and controlled from EPICS via the ColdFire CPU. Detector analog readout data is also sent over EPICS. Thus, the BLM LN can also be used as a radiation monitoring instrument.

PIC Link-Node

The Protection Ion Chamber LN is similar to the BLM LN. It interfaces the PIC loss monitors to the MPS. The output of the PIC chamber is connected directly to the LN. The chamber signals are digitized and processed using the same custom QADC module, operating in PIC mode. The LN can accommodate up to four IP-QINT-ADC modules, allowing for a total of 32 PIC chambers to be connected to a single PIC LN. This LN is a triggered device and also controls the PIC high voltage power supply chassis. Fault thresholds and analog data are setup and read out respectively over EPICS via the ColdFire.

ByKIK Link-Node

A single bunch beam dumper magnet, called ByKIK, is located upstream of the Undulator. This magnet interfaces to the MPS through the ByKIK LN. This LN receives Standby and Abort triggers from the timing system and provides a conditioned trigger to the kicker pulse generator, qualified with an MPS permit signal. The kicker outputs an analog waveform that indicates the quality of the kick pulse. This waveform, is digitized (using a COTS IP ADC) by the ByKIK LN and the sampled waveform is compared against thresholds. If the comparison exceeds these thresholds, an MPS fault is generated and the beam is disabled. Specially configured PIC L- and Interface Boards connect the kicker signals to the ADC.

LINK-NODE SOFTWARE

The LN code is EPICS running on the RTEMS RTOS. EPICS applications code is written in C. Information from the LN is available as EPICS Process Variables (PVs) displayed on specialized EDM panels. The PVs are read out of the LN at a 0.5Hz polling rate. Critical PVs are save/restored, archived and connected to the alarm handler.

OPERATIONAL EXPERIENCE

The LCLS MPS was commissioned in 2009. Currently, there are a total of 32 Link-Node chassis in the system. Prior to commissioning, an interim MPS was setup using legacy hardware. By summer of 2010, all of the legacy TIU and 1553-MPS inputs were moved over to LCLS MPS Link-Nodes; fully transitioning MPS functionality for LCLS to the LCLS MPS. The system has been running satisfactorily for almost two years now.

FUTURE DIRECTIONS

This same system will be used for machine protection in the upcoming LCLS-II machine. Plans exist to upgrade the LN ColdFire CPU with a more powerful COM Express platform unit. Based on recent needs, the design of two more flavours of LN are being considered: thermocouple input and general-purpose analog input units. The flexibility of the platform makes these latter items relatively easy to implement.

REFERENCES

[1] R. B. Neal, et al., SLAC Blue Book "The Stanford Two Mile Accelerator", chapter 21, section 21-4 "Equipment Protection Systems"; 1968

[2] S. Clark, et al., "Smart Machine Protection System" ICLAPECS 1991 Conference, SLAC-PUB-5688.

[3] S. Norum et.al., "The Machine Protection System for the Linac Coherent Light Source", PAC09, Vancouver B.C. Canada, FR5REP039. May, 2009.

[4] P. Krejcik, "LCLS Machine Protection System Requirements" SLAC/LCLS Internal Document, ESD 1.3-312-r1, March 2, 2006

[5] J. Dusatko et.al., "The LCLS Timing Event System", 2010 Beam Instrumentation Workshop, TUPSM083; May, 2010.

[6] J. Dusatko et.al., "The LCLS Beam Loss Monitor Readout System", 2010 Beam Instrumentation Workshop, TUPSM084; May, 2010.

STUXNET AND THE IMPACT ON ACCELERATOR CONTROL SYSTEMS

S. Lüders, CERN, Geneva, Switzerland

Abstract

2010 has seen wide news coverage of a new kind of computer attack, named "Stuxnet", targeting control systems. Due to its level of sophistication, it is widely acknowledged that this attack marks the very first case of a cyber-war of one country against the industrial infrastructure of another, although there is still much speculation about the details. Worse yet, experts recognize that Stuxnet might just be the beginning and that similar attacks, eventually with much less sophistication, but with much more collateral damage, can be expected in the years to come. Stuxnet was targeting a special model of the Siemens 400 PLC series. Similar modules are also deployed for accelerator controls like the LHC cryogenics or vacuum systems as well as the detector control systems in LHC experiments. Therefore, the aim of this presentation is to give an insight into what this new attack does and why it is deemed to be special. In particular, the potential impact on accelerator and experiment control systems will be discussed, and means will be presented on how to properly protect against similar attacks.

DAWN OF A NEW ERA

In June 2010, repeated reports of a major wave of infected Windows PCs in Iran hit the news. Initially triggered by a report of a Belarus security company called VirusBlokAda on a new Windows zero-day exploit*, a deeper analysis by Symantec, another security company, revealed that this exploit targeted specifically the SCADA (Supervisory Control And Data Acquisition) systems manufactured by Siemens.

Such SCADA systems have been deployed in thousands of instances worldwide, e.g. in the car manufacturing industry, in facility management, oil & gas industries as well as for accelerator control systems. However, in this particular instance, it seemed that this exploit targets the control system of the nuclear facility of Natanz in Iran. This plant is used for the enrichment of uranium. For Siemens, it was an unfortunate coincidence, that their systems were used there.

Named "Stuxnet", as derived from some keywords buried in the exploit code, this is the first documented exploit which deliberately attacks control systems. Due to its level of sophistication, there was much speculation whether this is a case of cyber-war of a particular country against the industrial infrastructure of another, namely the U.S. and Israel against the Natanz nuclear facilities in Iran [1]. Indeed, the sophistication of Stuxnet is very impressive, and this level of sophistication and complexity makes it most unlikely that Stuxnet was produced by an average attacker, but instead required significant investments, engineering skills and intelligence gathering. However, due to the nature of the attack, it is impossible to obtain confirmed information.

The media quickly labelled Stuxnet as "a new kind of cyber-attack" [2], but it was not. In the 1980's, the U.S. CIA provided a Russian energy provider with manipulated valves which eventually led to the explosion of a Siberian natural gas pipeline [3]. Also, the inherent weaknesses exploited by Stuxnet have been reported before, like the CERN TOCSSiC tests conducted in 2005 and later which found 39% of 35 tested devices being susceptible to malicious packages [4] and that many of these devices lack basic access protection allowing them to be manipulated and/or stopped easily. It took five more years for an ultimate proof.

THE INNER WORKINGS

The Stuxnet attack proceeds in two steps: in a first step Stuxnet infects random Windows PCs, in a second it attacks the controls process. A general overview of the different phases is given in Figure 1.

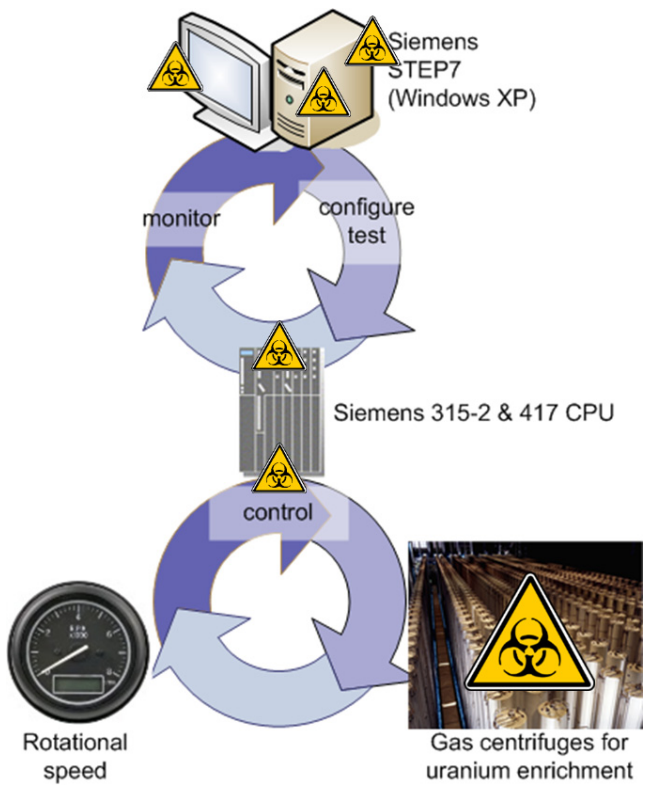


Figure 1: Sketch of a possible control system. Components which were attacked and compromised by Stuxet have been marked explicitly.

* A "zero-day exploit" is an exploit targeting a currently unknown vulnerability. Hence, no fix or patch exists.

Phase 1: Infect a SCADA PC

The PC attack was taking advantage of four (4!) initially unknown vulnerabilities in the Windows operating system. These vulnerabilities are quite valuable since they might be traded on the black market for more than $100 000 each [5]. Most likely, Stuxnet has been introduced via an infected USB stick by either a malicious insider, a saboteur or by means of social engineering luring an employee to insert the stick into a PC.

Once having successfully infiltrated the PC, Stuxnet hides itself using so-called root-kit functionality. This functionality takes benefits of two certificates stolen by the adversaries from a Taiwanese company. This gave the root-kit the air of legitimacy. This makes it neither visible to any local user who browses the infected program folders nor to any local anti-virus software. It then scans the local network and tries to infect further hosts. Using peer-to-peer functionality, all nodes are kept up-to-date and in contact with two remote command and control servers, one in Denmark, the second in Malaysia.

So far, Stuxnet behaves like a sophisticated and expensive worm harvesting compromised PCs and waiting for a command to unleash its malicious power. Up to five different versions with different functionalities have been identified, the oldest dating back to June 2009. In the end, Stuxnet has infected approximately 100.000 PCs worldwide (60% in Iran, 18% in Indonesia, 5% in India) [6].

Phase 2: Compromize the Control Process

But Stuxnet is special. Once established on a PC, it checks for the presence of the Siemens "STEP7", "WINCC" or "PCS7" software suites [7][8][9]. The STEP7 software is essential to program the control system in so-called programmable logic controllers or PLCs. The control system software consists of a series of software "blocks", e.g. "Function Blocks" (FB/FC), "Operational Blocks" (OB), or "Data Blocks" (DB) which are combined into a "project". WINCC and PCS7 are two types of Siemens SCADA applications displaying information relayed from the PLC and allowing values to be changed interactively in the PLC.

If one of these software suites is present, Stuxnet plants a copy of itself into any STEP7 project which can be found on the PC. This will open a further vector of propagation, should the project is copied and transferred to another PC by the corresponding system expert.

In addition, Stuxnet replaces the so-called S7 communication libraries (DLLs) such that it can fully control any data exchange between the SCADA PC and the PLC (see Figure 2). It acts now as a "Man-in-the-Middle" which can hide certain information from the operator looking at the SCADA display like out-of-bound values or alarms, and inhibit commands issued by an operator to the PLC.

Next, Stuxnet scans for PLCs which are reachable from that SCADA PC and "fingerprints" those PLCs: The PLC hardware must be of a certain type of hardware module and the software must contain a number of user-defined blocks with a certain byte-pattern and length. If the fingerprint does not match certain criteria, the worm stays idle and would expire on June 24th 2012.

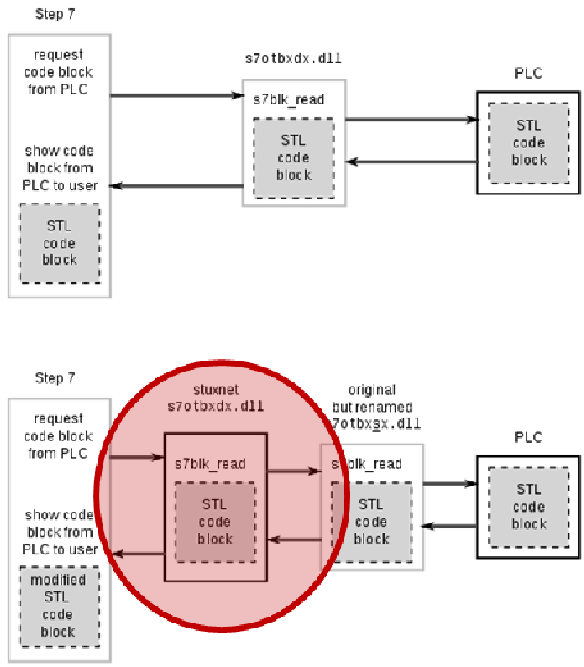


Figure 2: The Siemens S7 communication before (above) and after (below) Stuxnet has inserted its malicious libraries [10].

Phase 3: Game Over

Finally, if the control process matches what Stuxnet is looking for, Stuxnet downloads 17 to 32 additional software blocks to the PLC (depending on the PLC type). This new software blocks change the local control process such that, over the course of several months, the rotational speed of the gas centrifuges deviate from their nominal 1400Hz up to 1410Hz or down to a few hundred hertz [10]. Due to this, the uranium enrichment process was hampered, and the centrifuges wear out more quickly.

Due to the "Man-in-the-Middle" controlling the data flow from and to the SCADA system, operators in front of their SCADA displays will not have noticed anything. Thus, the production of highly enriched uranium became spoiled, and hence delayed.

PROTECTIVE MEANS

Stuxnet is targeting only one particular control system allegedly located in Iran. Therefore, given the aforementioned fingerprinting, it is very unlikely that accelerator or experiment control processes match these patterns. But in the course of the attack, Stuxnet also infects Windows (SCADA) PCs, which are employed in accelerator or experiment control processes, too.

However, not being affected *this* time should not imply that taking no action is an option. On the contrary, Stuxnet

should be considered as a wake-up call to raise the barriers, if not done yet, and deploy measures for properly protecting controls systems.

Protecting a Siemens PLC

While not broadly discussed in the media nor explicitly publicized by Siemens, their PLC come with two basic protection means: a local firewall and an intrusion detection system.

Sophisticated Siemens PLCs, as well as PLCs from other manufacturers, too, come with rudimentary firewall functionality based on IP access protection lists. For proper protection, only IP addresses of remote devices (PLCs, PCs) with a need to communicate with that particular PLC should be permitted by this firewall. All other connections should be dropped[†].

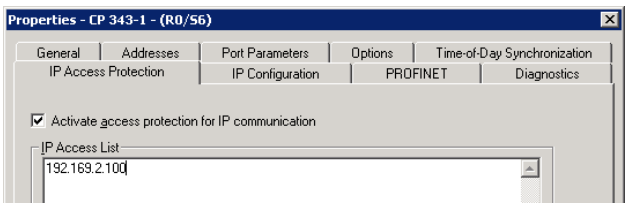


Figure 3: Configuration window of a Siemens S7-343 communication processor. The "IP Access Protection" tab allows configuring a local firewall based on IP addresses.

Special to Siemens STEP7 software is the provisioning of a so-called "Checksums" functionality which can be used as a simple intrusion detection system. Checksums are quasi-unique numbers ("hashes") which change drastically once a bit of the PLC's software blocks or its hardware is modified[‡]: STEP7 allows calculating such a hash over all software blocks, i.e. FB/FC/SFB/SFC/DB/OB, of a STEP7 project. A second hash is calculated over the hardware configuration. These values can be compared to those computed directly on-line by the PLC's CPU (see Figure 4).

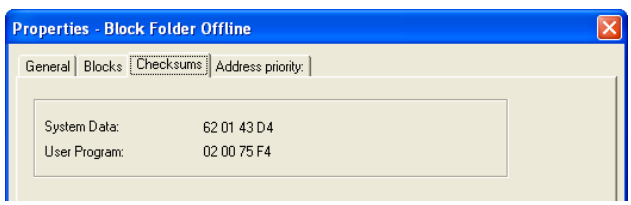


Figure 4: Information box giving the checksum values of a STEP7 project.

Mismatches indicate that software blocks in the PLC or its hardware has been altered. In such cases, the PLC or an attached display can raise an alarm (alarm e-mail, alarm sound, warning message). Important is, however, that the display can and is not compromised through a "Man-in-the-Middle" attack as the SCADA PC was[§].

Defence-in-Depth

The general strategy for the protection of control systems (as for any other computer system) must follow a "Defence-in-Depth" approach. This implies that protective measures have to be deployed on every level of the hardware and software stack[**]:

- On the network level by proper network segregation and compartmentalization of the controls network into security cells;
- On the hardware level by increasing the robustness and resilience of the device itself. Physical outlets like USB ports or CD drives should be disabled if not essential for operation;
- On the level of network services by disabling those services which are not essential for operation. A device should not be susceptible nor fail to a simple network scan issued by e.g. Nessus [11] or nmap [12];
- On the firmware and operating system level by applying software upgrades and patches in a timely manner. If possible, the device should run anti-virus software with up-to-date virus signature files. All default passwords must be changed and not disclosed to any other third party;
- On the application level by removing all applications which are no essential for operation. All remaining applications must be kept up-to-date and patched in a timely manner;
- On the social level by providing appropriate training to the system experts and operators. Particular focus should be put on awareness raising and the risks of social engineering[††].

More details on any "Defence-in-Depth" approach can be found in the "Good Practise Guidelines" of the U.K. CPNI [13] or in the ISA SP99 series of documents [14]. Other good standards are [15][16][17][18]. Usually, it is sufficient to choose any one standard or guideline and cross check later with a second one.

For example, long before Stuxnet, CERN set up a working group on the protection of control systems [19]. Since then, all control systems at CERN must adhere to the "Computer and Network Infrastructure for Controls (CNIC) Security Policy for Controls" [20] and to the

[†] However, since the Stuxnet comprimized PC was directly linked to the PLC, a local firewall wouldn't have prevented Stuxnet from connecting to that PLC.

[‡] Since this checksum is only four bytes long, there is a residual risk for finding collisions, i.e. the same checksums for a completely different set of software blocks or hardware configuration. However, to the author's knowledge, Stuxnet has not been designed to produce such a collision.

[§] There is still the risk of a "race-condition" where the malicious block downloaded to a PLC competes with the regular execution of the calculation of checksums. Depending on the exact time the malicious block gets enabled on the PLC, it might be able to inhibit raising an checksum alert.

[**] Admittedly, not all those measures would have helped to protect against Stuxnet. In particular, Stuxnet's zero-day exploits were not patchable at that time. Also, Stuxnet's sophistication bypassed the installed anti-virus software.

[††] Other good practices would require regular screening of experts and operators with view of their financial, social, family, and psychological situation, and whether they are addicted to drugs, gambling or alcohol. However, in the academic environment of high energy physics, this would be impossible.

"CERN Security Baselines" [21] which both give basic recommendations on how to protect control systems as well as computing services at CERN.

Whilst following a standard is definitively a good practise, it also must be stressed that any investment in cyber-security must be balanced. Consciously accepting the risk is a valid alternative, too.

CONCLUSIONS

Stuxnet should have been the wake-up call for all those who never believed that control system could and would be attacked. Indeed, the media was quickly interested and created an unprecedented hype about the protection of the critical infrastructure which strongly relies on control systems. They called out the new "Era of Cyber-War" [1]. In parallel, attackers as well as security researchers became interested in analysing control system hardware. Standard IT security companies joined and now provide technical solutions for protecting control systems, even if their technical knowledge in control systems might be limited.

Most importantly, vendors are also concerned. Siemens, for example, has started an elaborative initiative on securing their control systems [22]. This should, however, not serve as an argument to users not to take any action. On the contrary, users should review the security protections put in place. Deploying a "Defense-in-Depth" approach is mandatory and corresponds to good practise. Continuing to ignore control system cyber-security is grossly negligent.

A new era has begun. Stay tuned.

REFERENCES

[1] Spiegel Online, "Stuxnet Virus Opens New Era of Cyber War", August 8, 2011; http://www.spiegel.de/international/world/0,1518,778912,00.html.

[2] The Economist, "The meaning of Stuxnet", September 30, 2010; http://www.economist.com/node/17147862.

[3] The Washington Post, "CIA slipped bug to Soviets", February 27, 2004.

[4] S. Lüders, "Control Systems Under Attack?", ICALEPCS, Geneva, October 2005.

[5] C. Miller, "The Legitimate Vulnerability Market", May 6, 2007; http://weis2007.econinfosec.org/papers/29.pdf.

[6] Symantec Corporation, "W32.Stuxnet Dossier", February 2011; http://www.symantec.com/en/ca/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

[7] Siemens AG; http://www.automation.siemens.com/mcms/simatic-controller-software/en/step7/pages/default.aspx.

[8] Siemens AG; http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/Pages/Default.aspx.

[9] Siemens AG; http://www.automation.siemens.com/mcms/process-control-systems/en/distributed-control-system-simatic-pcs-7/pages/distributed-control-system-simatic-pcs-7.aspx.

[10] Wikipedia,org; http://en.wikipedia.org/wiki/Stuxnet.

[11] Tenable Network Security, "Nessus Open Source Vulnerability Scanner Project"; http://www.tenable.com/products/nessus.

[12] Fyodor Vaskovich, "nmap — Free Security Scanner for Network Exploration and Security Audits"; http://nmap.org.

[13] U.K. Centre for the Protection of the National Infrastructure (CPNI), "Process control and SCADA security - good practice guidelines"; http://www.cpni.gov.uk/advice/infosec/business-systems/scada/

[14] The International Society of Automation (ISA), "Manufacturing and Control Systems Security", ANSI/ISA SP99 TR99.00.01-04 http://www.isa.org/MSTemplate.cfm?MicrositeID=988&CommitteeID=6821.

[15] U.S. Department of Homeland Security (DHS), "Cyber Security Procurement Language for Control Systems", 2009; http://www.us-cert.gov/control_systems/pdf/FINAL-Procurement_Language_Rev4_100809.pdf.

[16] U.S. National Institute of Standards and Technology (NIST), "Guide to SCADA and Industrial Control Systems Security", NIST SP800-82; http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf. .

[17] U.S. Federal Energy Regulatory Commission (FERC), "Critical Infrastructure Protection CIP-002 to CIP-009"; http://www.nerc.com/page.php?cid=2%7C20.

[18] International Organization for Standardization (ISO), "Information Technology — Security Techniques", ISO/IEC 27001:2005 and following.

[19] U. Epting et al., "Computing and Network Infrastructure for Controls", ICALEPCS, Geneva, October 2005.

[20] S. Lüders et al., "CNIC Security Policy for Controls", 2011; https://edms.cern.ch/document/584092.

[21] S. Lüders et al., "CERN Security Baselines"; http://cern.ch/security/rules/en/baselines.shtml.

[22] ARC Advisory Group, "Risk Drives Industrial Control System Cyber Security Investment", May 2011; http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/ARC-Siemens-CyberSecurity-2011-v1.pdf.

DEVELOPMENT OF THE DIAMOND LIGHT SOURCE PSS IN CONFORMANCE WITH EN 61508

M.C. Wilson, A.G. Price, Diamond Light Source, Oxfordshire, UK.

Abstract

Diamond Light Source is constructing a third phase (Phase III) of photon beamlines and experiment stations. Experience gained in the design, realization and operation of the Personnel Safety Systems (PSS) on the first two phases of beamlines is being used to improve the design process for this development. Information on the safety functionality of Phase I and Phase II photon beamlines is maintained in a hazard database. From this, reports are used to assist in the design, verification and validation of the new PSSs. The data is used to make comparisons between beamlines, to validate safety functions and to record documentation for each beamline. This forms part of a documentation process demonstrating conformance to EN 61508 [1].

INTRODUCTION

The Diamond Light Source is a 3rd generation synchrotron, a particle accelerator that accelerates electrons close to the speed of light and uses insertion devices to cause the electron beam to produce intense beams of X-rays. It is used for research by the science community and industry. Inherent in the process is the risk of harm to personnel from ionising radiation, i.e. X-rays generated by design and neutrons and gamma radiation generated by collisions of high energy electrons with hard surfaces. As a result Diamond is subject to the UK's Ionising Radiation Regulations, IRR99 [2] and must provide an engineered safety system to manage the hazards. Diamond elected to adopt the safety system standard EN 61508 to guide the process. It is a generic international standard for the design of electrical / electronic and programmable electronic safety systems, which is applicable to environments without specific safety standards of their own, such as accelerators. Diamond has now been producing Personnel Safety Systems (PSS) to protect personnel from ionising radiation and other hazards since the project began in 2002. Protection against "Other hazards" may include crush hazards caused by robots, eye damage from lasers and any additional hazards which require a rigorous access control system.

The Diamond PSS includes:

- a Linac zone,
- 3 booster zones,
- 10 storage ring zones,
- 7 phase 1 beamlines
- 14 phase 2 beamlines

Phase 3 will consist of a further 10 beamlines.

Each system is an independent safety system which contributes to the overall safety system for the facility. This type of architecture allows for the addition of new systems and the removal of old without undue reconfiguration of the overall system. However, this amounts to a lot of safety systems to manage. As part of the safety management under EN 61508, a database of hazards was established. This has proven to be of little direct benefit, and so it was decided to extend its function to help manage the EN 61508 process. This paper describes the database and its role in the Diamond Personnel Safety System.

EN 61508 IMPLEMENTATION

Diamond identified the desire to apply EN 61508 to the PSS early in the realisation of the project. An external contractor was employed to set up the process and, due to pressures of the programme, the first iterations of the design and conformance processes were established at the same time. Diamond safety management requires formal identification of hazards in a HAZard IDentification process (HAZID). The technique, in common with many others, identifies the hazards that fall within a specific area and for each hazard the initiating event that might lead to an incident, the consequences, the likelihood of the incident and the safeguards that will protect against it.

The information collected in the HAZID is used to generate a safety requirements specification and as a basis of a safety model.

The safety requirements specification lists and describes the safety measures required and allocates them to subsystems, where appropriate, using Safety Integrity Levels (SIL), derived from the safety model. SIL indicate the rigour required from the safety system with 1 the least and 4 the most rigorous requirement. The system is built and tested so that each safety requirement is tested against a functional performance test. The remaining EN 61508 implementation concerns the quality of the process, involving verification and validation and cross-checks to ensure what was built meets the SIL requirement and is as designed, that the design is as specified and that the specification mitigates the hazards.

DATABASE OBJECTIVES

Diamond has an EN 61508 process which augments a traditional manufacturing process used to design, build and test the PSS. The EN 61508 process provides the means of managing the design and manufacture; however the manufacturing process is typical of manufacturing processes used elsewhere without the requirement for EN 61508. Hence, there was a possibility that the two processes could operate largely independently and on different timescales, undermining the safety design process. A means of consolidation was therefore sought, such that the manufacturing process should be able to

receive timely information from the EN 61508 process in a readily useable form. The EN 61508 should be able to accept data from the design process and provide cross checks to ensure that the design is complete. The EN 61508 process should be able to provide calculations which support the design prior to the detailed analysis being undertaken.

There are three main elements that make these requirements appropriate to a database:

- the large number of similar and comparable systems at Diamond,
- the quantisation of qualitative information,
- the ability to structure reports based on a common data set in many different arrangements for cross referencing and validation of the various stages of EN 61508

Having a large number of systems allows the comparison of data between systems. This is a valuable benefit, as the start point of the process requires human expertise and judgement, which is fallible. Cross referencing between systems helps to identify omissions and irregularities by requiring justification of the differences.

Entering information into a database provides the opportunity to put similar interpretations and numbers to qualitative assessments. This is the first stage of processing data to provide probabilistic assessment of risk.

Producing reports that contain the pertinent information in an appropriate order is a great help to verification and validation. Information from different points in the process can be brought together to generate new information and data, including calculations.

DATABASE STRUCTURE

The database is built using Microsoft Access 2007 from the Microsoft Office suite. It consists of tables of hazards, safeguards, frequencies of opportunity and outcome severities. These are linked to interlocks and control measures by a logic table. Preliminary assessments can be undertaken and reports generated to support the EN 61508 process. Documentation is also included by linking the database into the SharePoint document repository. The following sections identify the processes and reports generated for the various stages.

HAZARD IDENTIFICATION

The Diamond Safety Management plan requires hazard identification to be undertaken early in the project. As soon as the preliminary beamline layout has been established, it is reviewed for hazards, and the potential frequency and severity of incidents is assessed. Safeguards to prevent the incident are identified. The results are recorded in the "Preliminary Hazard Analysis" report. The records of the review are collated and recorded in the database. In entering the information, steps are taken to convert the qualitative results into quantitative values:

- Frequency of opportunity is commonly expressed as an interval during discussion and is converted into an explicit number of events per annum.
- Severity is allocated by table to provide a risk of fatality per event as a percentage. This provides common allocation of severity for similar hazards on different systems.
- The effectiveness of the safeguard is assessed and probability is assigned to the control measure.

The first output from the database is a report for checking the database records against the Preliminary Hazard Analysis report. The report is formatted to present the data in the same arrangement as the HAZID table for ease of comparison

"Sore Thumb" Checks

The benefit derived from implementing a system to structure the design of a safety system can only be as good as the identification and analysis of the hazards presented. At Diamond we have the benefit of having built many safety systems and have records of the hazards identified on those systems. The database is used to generate two reports that compare hazards, frequency, severity and safeguards, in different formats, allowing the comparison and rationalisation of differences between hazards on different beamlines. The first format arranges the initiating event with consequence and frequency of opportunity grouped by beamline and beamline area. This allows comparison of hazards identified in similar areas on different beamlines. The second format arranges the data by cause, which generates a report that allows comparison of the treatment of a particular hazard on different beamlines.

Hazard and Safeguard Cross Reference

Once the appropriate and correct data is in the database a cross-reference report is generated and appended to the Preliminary Hazard Analysis report.

PRELIMINARY NUMERICAL ANALYSIS

Undertaking some preliminary analysis is a useful exercise, as iterations "around the EN 61508 loop" are quite time consuming.

Risk of Fatality Report

Given that most safeguards contribute to additional safety, a very basic preliminary calculation can be undertaken that combines the safeguard contributions as a logical AND function of all the individual safeguards that contribute to the mitigation of the risk. It should be noted that accuracy of this calculation is limited by the simplicity of the model, which ignores the independence and common-mode failures of the safeguards, and does not replace the need for a more thorough analysis. However it does provide a useful early warning that more

safety measures may be required to provide adequate mitigation for a particular hazard.

SIL Rating Report

Most of Diamond's safety systems have similar safety requirements and similar solutions; this allows the use of common analysis between safety systems. It is not unreasonable to assume that an equivalent system on similar safety systems will have the same SIL requirements and that the equivalent solution will achieve a similar SIL. The database also holds references to the evaluation of the solution.

The SIL report provides a list of SIL evaluations, SIL ratings and the Probability of Failure on Demand (PFD) for the function. This allows the development of safety models using consistent values.

SAFETY REQUIREMENTS SPECIFICATION

The safety requirements specification contains definitions of the safety functions that the system will implement. It is derived from the safeguards and control measures defined in the database. The first report in this section lists the safety requirements for inclusion in the specification. It includes legal requirements and Diamond safety policy functional requirements, which may not be direct "safety" requirements placed upon the safety system. It should be noted that the report lists an occurrence of a particular safety function for each hazard that it mitigates. The safety function is incorporated only once in the design as it is effective for all of the hazards it mitigates, but the complete list of references is an important record in case of future changes of requirements.

The list of safety functions produced from the database is a significant element in the composition of the safety requirement specification, which is the document that specifies in detail the function of the personnel safety system and its components.

SAFEGUARD COMPARISON REPORT

Again, as we have a history of safety systems design at Diamond, it is prudent to compare the current system with previous similar systems. A report listing the safeguards for particular hazards allows comparison and justification of differences between beamlines and assumptions made at the time of hazard identification.

NON-PSS SAFETY REQUIREMENTS REPORT

Some of the safety functions required to achieve adequate safety fall outside the scope of Electrical / Electronic / Programmable Electronic (E/E/PE) systems and are managed by others. This report lists all Non-PSS safety requirements. Its use provides an effective mechanism for communicating and discharging responsibilities.

LOGIC DESIGN

Safety functions translate into interlocks and control measures, or "permits". Often, the same control measures are required for several interlocks, and so logic functions are required to combine interlocks and provide permits to equipment. Generally the control measure is asserted only if the logical AND combination of the interlocks is present. By processing the beamline data in the database to produce a report of interlocks by control measure, a comprehensive design-checking tool becomes available. The database provides a method of recording the logic design so that each safety function is implemented and is recorded.

VERIFICATION AND VALIDATION

The database allows the safety functions to be listed for the each system. A report can be generated which lists each safety function, its interlock and its permit. This is a valuable tool for checking that all safety functional requirements are tested in the functional test procedure. Further, the reference for the functional test for each safety function can be recorded to provide a cross-check that all safety functions have been included and are tested.

DOCUMENTATION

The documents associated with a beamline are referenced from the database into the document repository. This allows generic documents and system-specific documents to be used for each safety system and the documents to be recalled simply and efficiently.

CONCLUSIONS

The ability of the database to produce reports sorted and filtered on different criteria allows the production of reports tailored for specific stages of the EN 61508 process. Presenting similar data in different formats enhances the ability to check and compare data. Cross referencing and filtering on specific criteria provides valuable information for assisting audits. The database is a useful tool for checking validity, assisting design and verifying the safety system, but should not be employed blindly.

FUTURE DEVELOPMENT

The database contributes to the processes involved with the development of safety systems, during which assumptions are made about the frequency of operation and reliability of components in the system. The database could be developed to accommodate data collected from operational experience allowing the assumptions to be validated. In practice, frequency-of-operation information is relatively simple to collect; however there are often other considerations to be taken into account and some sort of normalisation may be required before the number becomes useful. Reliability of components is a study that is likely to be too complex for a database of this type, and

is probably better studied elsewhere. Consolidation of failure data into subsystem failure and safety requirement failure would be of interest. Safety requirement failure rates could be added to the database for validation of SIL ratings. Subsystem failure data could be collected elsewhere for comparison with Failure Modes, Effects and Consequence Analysis (FMECA) and subsequent studies.

REFERENCES

[1] EN 61508, "Functional Safety of Electrical/ Electronic/Programmable Electronic Safety-related Systems"
[2] IRR99, "The Ionising Radiation Regulations, 1999"

CENTRALISED COORDINATED CONTROL TO PROTECT THE JET ITER-LIKE WALL*

A.V. Stephen, G. Arnoux, T. Budd, P. Card, R. Felton, A. Goodyear, J. Harling, D. Kinna, P. Lomas, P. McCullen, P. Thomas, I. Young, K-D. Zastrow, EURATOM-CCFE Fusion Association, Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK
A. Neto, D. Alves, D.F.Valcárcel, Associação EURATOM/IST, Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, 1049-001 Lisbon, Portugal
S. Jachmich, Association 'EURATOM - Belgian State' Laboratory for Plasma Physics Koninklijke Militaire School - Ecole Royale Militaire Renaissancelaan 30 Avenue de la Renaissance B-1000 Brussels Belgium
S. Devaux, Max-Planck-Institut für Plasmaphysik, EURATOM-Assoziation, D-85748 Garching, Germany
and JET EFDA Contributors*, JET-EFDA, Culham Science Centre, OX14 3DB, Abingdon UK

Abstract

The JET ITER-like wall project (ILW) replaces the first wall carbon fibre composite tiles with beryllium and tungsten tiles which should have improved fuel retention characteristics but are less thermally robust. An enhanced protection system using new control and diagnostic systems has been designed which can modify the pre-planned experimental control to protect the new wall. Key design challenges were to extend the Level-1 supervisory control system to allow configurable responses to thermal problems to be defined without introducing excessive complexity, and to integrate the new functionality with existing control and protection systems efficiently and reliably. Alarms are generated by the vessel thermal map (VTM) system if infra-red camera measurements of tile temperatures are too high and by the plasma wall load system (WALLS) if component power limits are exceeded. The design introduces two new concepts: local protection, which inhibits individual heating components but allows the discharge to proceed, and stop responses, which allow highly configurable early termination of the pulse in the safest way for the plasma conditions and type of alarm. These are implemented via the new real-time protection system (RTPS), a centralised controller which responds to the VTM and WALLS alarms by providing override commands to the plasma shape, current, density and heating controllers. This paper describes the design and implementation of the RTPS system which is built with the Multithreaded Application Real-Time executor (MARTe) and will present results from initial operations.

INTRODUCTION

The main experimental control of the JET[1] machine is provided by five real-time control systems, or actuators, which govern : plasma position and current control (PPCC [2]), fuelling, via the plasma density local manager (PDLM) and additional heating, via the local manager systems for neutral-beam heating (NBLM), radio frequency heating (RFLM) and lower-hybrid current drive (LHLM).

JET machine protection has been provided historically by three systems. The Central Interlock and Safety System (CISS) provides basic hardwired plant protection. Higher level protection with limited configurability is provided by the Pulse Termination Network (PTN[3]). This is triggered if key systems fail, or if protection signals move outside of predefined limits. The PTN output is a latched stop signal to each of the control systems, which execute a fixed shutdown sequence in response. Finally, additional heating systems require enable signals from the Plant Enable Window System (PEWS) which are conditioned by real-time protection signal validation algorithms.

The design of these systems is characterised by a high degree of configurability in respect of detecting machine fault conditions, but extremely limited control over stop strategies which are constrained to be fixed throughout the duration of an experiment. The stops are designed to minimise the global damage to the machine but are unable to take account of any localised heat flux limits on plasma facing components (PFCs).

The ILW project[4] has rebuilt the plasma facing surfaces of the machine with all-metal tiles which should have improved fuel retention properties. However, their temperature needs to be carefully controlled to avoid damage. The Protection of the ITER-like Wall (PIW) project was launched in late 2009 to upgrade the protection systems to address this need.

CONCEPTUAL DESIGN

The PIW system design aimed to augment the diagnostic monitoring of wall component temperatures and to provide a new protection system which could intervene to keep these temperatures within limits. The new system should act early, to prevent PTN or CISS from following a global stop strategy that might cause excessive energy loads on the walls. However, the basic machine protection provided by PTN and CISS had to be retained. The structural archi-

* See the Appendix of F.Romanelli et al., Proceedings of the 23rd IAEA Fusion Energy Conference 2010, Daejeon, Korea.

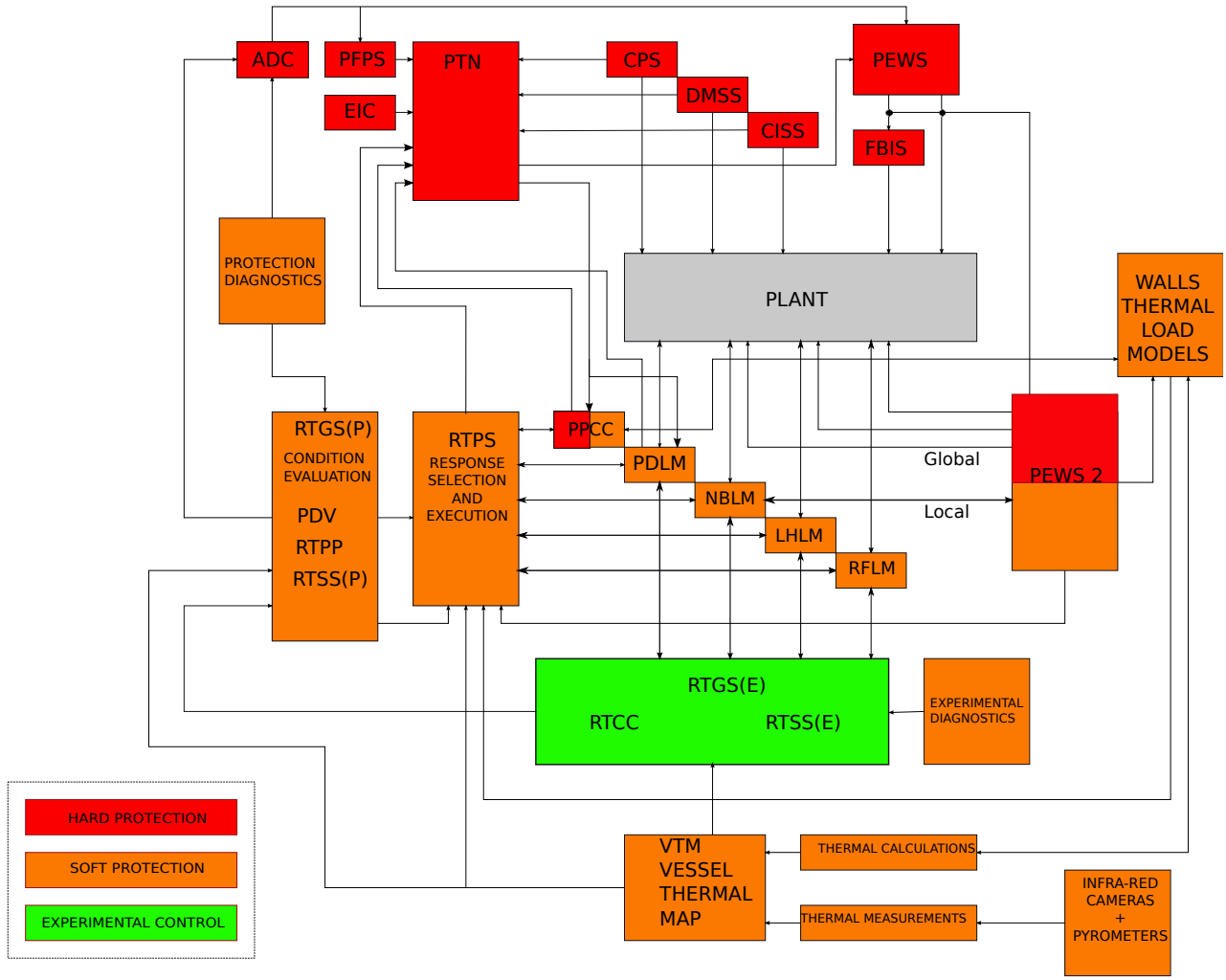


Figure 1: JET protection of the ITER-like wall system architecture. The experimental real-time control links to the density and heating local managers (PDLM, NBLM, RFLM and LHLM) is expanded to allow protection overrides from the new RTPS system which receives alarms from VTM, WALLS and also from the real-time plasma protection (RTPP) and real-time central controller (RTCC) systems. The original higher level protection systems of CISS, PTN and PEWS are retained.

tecture was clear and is outlined in figure 1. The density and additional heating real-time controllers already had interfaces which allowed for real-time modification of plant control. PPCC would be similarly updated. A new system, called the Real-Time Protection Sequencer (RTPS) would provide protection override commands to adapt the experimental control as required to keep temperatures under control, or to achieve a safe termination when necessary.

Temperature monitoring is provided based on real-time measurements from new infra-red camera and pyrometer diagnostics[5] which are analysed by a new system called the Vessel Thermal Map (VTM[6]) in order to generate alarms where limits are reached. The plasma wall load system (WALLS) may also generate alarms if modelled predictions of temperature or energy load become excessive. WALLS required to be updated to take account of the new component geometry and materials.

The design distinguishes between *local* and *global* events. Local alarms are limited in location or time, or may be set at lower thresholds and trigger a protection response which allows continued operation with dynamically controlled limits on further injection of heating with fine granularity. If this fails to bring checks back into tolerance, global alarms are raised which result in control overrides which truncate the experiment and land the plasma safely.

The significant design challenge was to make the new system practical to configure. The best control to provide a soft landing in the event of a fault condition depends on the state of the plasma. This is a function of the experiment design, the phase within a particular discharge, and the type of fault detected. Potentially, this represents a large number of possible paths. This was addressed with a fault and response classification scheme which confines the complexity to a manageable level, and by providing a way to define response actions generically, such that a library of reusable stop responses may be applied to many similar experimental scenarios.

RTPS IMPLEMENTATION

Local Protection

Local alarms and local protection are defined in relation to the additional heating systems. Each heating system comprises several individual units, and under certain conditions, particular heating units may directly heat a specific plasma facing component. For example, the neutral-beam heating system has 16 positive ion neutral injectors (PINIs). Depending on the plasma density, a proportion of beam particles pass through the plasma and strike the surface of the machine, a condition known as shine-through. Each PINI has a shine-through footprint on a specific set of components. Therefore, when a local hotspot alarm occurs for such an element, the relevant PINI should be turned off to prevent further overheating. This should not preclude the neutral-beam system as a whole from continuing to deliver the total requested power to the plasma, as other PINIs can be turned on to compensate. Similar relationships and control rules can be determined for the RF antennae, and the LH klystrons.

Global Protection

If local protection is not adequate to keep the machine safe the experiment must be curtailed. To manage the potential complexity of the design, the key observation is that although there are many hundreds of individual fault conditions, most require similar reactions, and we introduce the term *stop trigger* to classify the types of threat.

Seven stop triggers have been identified. Three represent thermal problems. These are hot spots in the main chamber (inner and outer walls), in the divertor (the base region where particles are exhausted from the machine) or which occur in both regions. Two are associated with magneto-hydrodynamic (MHD) instabilities. Two more allow for generic conditions requiring either a slow or fast termination of the pulse.

Each stop trigger is associated with a corresponding *stop response*, which defines the required override control strategy to mitigate the effects of the related alarm. There are three kinds of stop response. If a fault occurs early in the discharge before main heating phase, it can be sufficient to trigger the PTN system. During the heating phase, the real-time controllers require overrides which are fully programmable, termed *RTPS* stops. In some cases, the best possible way to end a pulse is to execute the control waveforms that would have come into force had the experiment reached a natural conclusion. This is termed a *jump-to-termination* (JTT) stop. Table 1 illustrates the matrix that links stop triggers to stop responses, as a function of experimental phase. The full control matrix is a key part of the protection system configuration interface.

The system also takes account of the possibility that one class of fault (and corresponding stop response) might be followed soon after by another. For example, a magnetic instability followed by a hotspot, or vice versa. This is addressed, by allowing for two levels of stop response, *primary*, and *secondary*. Not all possible combinations of control are allowable, since in some cases, once a primary stop response has begun, the best outcome is achieved by allowing it to run to completion.

Table 1: The primary stops table configures the mapping between stop triggers and stop responses as a function of experimental phase. A subset of the possible stop triggers are shown, including mode lock (MHD), main chamber hotspot (MCHS) and divertor hotspot (DHS).

Phase	Slow	MHD	MCHS	DHS
Breakdown	PTN	None	None	PTN
Ip Rise	|	|	|	|
Limiter	PTN	|	|	|
X-point	PTN	|	|	|
Heating 1	RTPS	|	RTPS	PTN
Heating 2	RTPS	|	RTPS	JTT
Plasma Termination	PTN	|	PTN	PTN

Stop Responses

A particular stop response is defined by a set of override references for each of the five actuator systems, each of which has one or more control signals. For a given control signal, the override is specified as a waveform which is parameterised as a set of steps, where for each step, the reference value to achieve and the time in which to make the transition are defined.

There are many ways of specifying these parameters. Reference values can be defined in absolute terms, or relative to the control value at the beginning of the stop, or in proportion to any other control signal. e.g. If 10MW of neutral-beam total power were being delivered and a stop response was initiated, the new reference could be set to 6MW (absolute mode), 60% (relative mode) or proportional to plasma current, in which case the reference would be calculated dynamically. Step durations can be defined as an absolute time, as a ramp rate (where the duration will be computed from the difference between the reference values) or to be synchronised to another control signal.

Software and Hardware

To achieve the required degree of configurability and reliability, the software is constructed using a component based approach, based on the Multi-threaded Application Real-Time executor (MARTe[7]). A MARTe application is organised as a chain of processing objects instantiated from Generic Application Modules (GAMs) interconnected by a software bus called the dynamic data buffer (DDB). A supervisory level set of tasks implements a state machine

distinguishing background operation, preparation for an experiment (pulse), pulse on, and post-pulse actions (including data collection). The set of GAMs and their attributes are data driven, which allows a powerful separation of source code and binary modules from experimental parameters. This allows the application to be comprehensively configured from the JET Level-1 system.

The standard MARTe library provided the key infrastructure, driver modules for the peripheral electronics and data collection modules. The RTPS software was completed by writing custom MARTe modules. These include the *Stop Selector* GAM which implements the state machine and alarm processing logic, and the *Stop Manager* GAM which defines and executes the stop responses. The application is executed on a PowerPC processor hosted in a VME rack, with digital IO modules providing links to the PTN system.

MARTe has been used with several operating systems and was ported to VxWorks 6.8 to develop the online real-time version of the RTPS application. A major benefit of using MARTe was the possibility of recompiling the application to run on Linux to allow development and offline testing with access to powerful debugging and checking software tools.

Communications with the alarm generator and control actuator systems is via the JET real-time data network (RTDN[8]) which provides low latency, high reliability exchange of fixed size datagrams over permanent virtual circuits. The processing rate is 500Hz, with inputs from the VTM at 100Hz.

Ensuring Reliability

Alarm systems such as the VTM define blind stop alarms to handle loss of critical signals during a shot. If RTPS detects communication or status faults with the alarm source systems, or real-time controllers, it can trigger the PTN system. A hardware watchdog signal to PTN ensures that RTPS is operational itself. Detection of PIW system failures needs to be reliable and appropriate. The Level-1 user interface provides intelligent conditioning of these checks so that features or subsystems which are not in use cannot cause problems.

The PIW system overall relies on distributed interaction between simpler, smaller systems. This reduces the cost of developing, commissioning and testing of each subsystem, but demands correct communications. To ensure this, all messages between systems are defined in a central database. Each message is allocated a unique identification number which is used to allow run-time verification that all systems on a virtual circuit are coherent and operating correctly.

OPERATIONS

JET recommenced operations with the ILW on 24 August 2011 and initial plasma performance has exceeded expectations. Unusually following a JET shutdown, the experimental and restart commissioning activities are interleaved. The restart programme has included sessions dedicated to the commissioning of the PIW systems.

The basic protection logic has been tested thoroughly, and the response of the PPCC and PDLM systems for a number of stop response types has been verified. End to end testing of alarm generation from camera systems through to RTPS response has been demonstrated. This was done initially without heating the wall, by using the cameras without infra-red filters. Calibration parameters and the visible plasma light intensity levels were tuned to simulate high temperature signals. Similar tests will be repeated using real temperature signals when the infra-red filters have been fitted, and the additional heating systems commissioned. RTPS has been used successfully to override a plasma discharge by requesting PPCC and PDLM to execute the jump-to-termination response in JET pulse number 80500.

CONCLUSIONS

The impact of plasma fault conditions which risk damage to the JET ITER-like wall will be mitigated by the action of a new, highly flexible protection system. This allows safer and more efficient exploitation of the new machine by expanding the operational space available, and by gently landing the plasma when necessary, without making the task of session preparation excessively onerous.

ACKNOWLEDGEMENTS

This work was supported by EURATOM and carried out within the framework of the European Fusion Development Agreement. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1] JET EFDA Contributors, “Special Issue on Joint European Torus (JET)”, Fusion Science Technology, Vol. 53, No. 4, pp.861-1227, 2008.

[2] Neto, A. C. et al., “Shape Controller Upgrades for the JET ITER-like Wall”, this conference.

[3] J. How et al., “The JET Pulse Termination Network”, Fusion Technology, Vol. 2, pp.1421-1426, 1980.

[4] G. Matthews et al., “Overview of the JET ITER-like Wall Project”, Fusion Engineering and Design, Vol. 85, issue 7–9, pp.1581–1586, 2010.

[5] M. Jouve et al., “Real-time protection of the ITER-like Wall at JET.”, this conference.

[6] Alves, D. et al., “The Software and Hardware Architectural Design of the Vessel Thermal Map Real-Time System in JET” , this conference.

[7] Neto, A. C. et al., “MARTe: A Multiplatform Real-Time Framework”, IEEE Transactions on Nuclear Science, 57 (2010), p.479–486.

[8] Sartori, F. et al., “JET Operations and Plasma Control”, Fusion Engineering and Design, Vol. 66-68, pp. 785-790, 2003.

HIGH-INTEGRITY SOFTWARE, COMPUTATION AND THE SCIENTIFIC METHOD

Les Hatton*, CISM, Kingston University, UK

Abstract

Computation rightly occupies a central role in modern science. Datasets are enormous and the processing implications of some algorithms are equally staggering. With the continuing difficulties in quantifying the results of complex computations, it is of increasing importance to understand its role in the essentially Popperian scientific method. In this paper, some of the problems with computation, for example the long-term unquantifiable presence of undiscovered defect, problems with programming languages and process issues will be explored with numerous examples. One of the aims of the paper is to understand the implications of trying to produce high-integrity software and the limitations which still exist.

Unfortunately Computer Science itself suffers from an inability to be suitably critical of its practices and has operated in a largely measurement-free vacuum since its earliest days. Within CS itself, this has not been so damaging in that it simply leads to unconstrained creativity and a rapid turnover of new technologies. In the applied sciences however which have to depend on computational results, such unquantifiability significantly undermines trust.

It is time this particular demon was put to rest.

BACKGROUND

High-integrity systems are fundamentally those systems which on failure, have a significant impact on humans, organisations, society or the environment. It is traditional to split these into two categories

- Safety-critical systems. Such systems on failure, have a direct impact on human safety or indirectly by damaging the environment. Examples of these include aircraft active avionics, railway signalling software or medical imaging software.
- Mission-critical systems. Such systems on failure, are typically associated with financial loss. In an organisation this could be failure of its accounting systems, customer information or other enterprise systems. However, in the context of science, such systems on failure would be directly associated with unintentionally misleading results. Indirectly, this may lead to financial loss but the essential issue here is untrustworthiness of the results.

In the context of Accelerator and Large Experimental Physics Control Systems, both categories will be represented. The failure of Control Systems is frequently safety-critical whilst data acquisition and processing is mission-critical in that failure here may well lead to misleading results.

The processes involved in producing Safety-Critical Systems are well-rehearsed, well-documented by numerous standards, both military (DO-178B, DO-254, DEF STAN OO-55/56) and civilian (CENELEC EN 50126, EN 50128, EN 50129, IEC 61508) and strongly associated with legal liability. In this paper, therefore, I will restrict myself to the latter of these issues, viz. the risk of unintentionally misleading results and the potential impact on scientific research.

SOME ISSUES IN HIGH-INTEGRITY SOFTWARE

The Scientific Method

The highly influential 20th century philosopher Karl Popper was instrumental in delineating the scientific method by laying down the notions of deniability. Deniability in an experimental context is fundamentally linked to openness of technique and reproducibility of result. We make progress by ruthlessly eliminating those experimental results which cannot be reproduced within some suitable bound of error when an identical experiment is performed. Such methods have served science well and have come to embody the very essence of the scientific method.

I will now restate deniability in a software context as

- Truth cannot be verified by software testing, it can only be falsified,
- Falsification requires quantification of computational modelling error,
- Deniability is at the heart of progress in scientific modelling. We are always seeking to deny the truth of a result and continued failure simply adds weight to a result but not verification,

In this context, Computer Science has not done well.

Software Defect

The principle enemy of the programmer is software defect. In the absence of a well-defined set of categories, I will define a defect as a fault or mistake in the software which then causes the software to fail. Software failure occurs when the behaviour of a computer program departs from its expected behaviour. Every failure is associated with at least one fault but not all faults fail.

There are many kinds of fault and there are many kinds of failure.

* lesh@oakcomp.co.uk

Fault categories Although by no means exhaustive or independent, these might include

- The use of ambiguous programming language features as described for example by [4],
- Mistakes in logic, for example an incorrect programming branch,
- Errors of omission, (i.e. something important left out such as initialisation statement),
- Errors of commission, (i.e. something put in which should not be there),
- Various abuses of floating-point arithmetic, [11],

Fault occurrence rates have been thoroughly examined by many researchers, for example the seminal contributions of Basili, [2]. Such faults are typically identified as occurring in the range 1-10 per thousand executable lines of code for typical software with arguably the best software around 0.1 on this scale.

Where computer scientists have made little progress is on *quantifying the effects these faults have on the actual output values of the computation, in other words on how they fail.* Let me then consider different kinds of failure.

Failure categories Examples of these might include

- An unintended program halt. These can occur for many reasons in programming languages. For example, an incorrectly dereferenced pointer or a pointer alignment problem in languages like C and C++; a divide by zero which causes a run-time exception; overflows; underflows and so on. Such errors although traumatic for a safety-critical system because service is interrupted perhaps during a critical phase, they tend not to be particularly serious in scientific computation because the scientist is immediately alerted that there is something wrong.
- Wildly incorrect results. These are results which are so far wrong that they cannot be correct as for example the rather surprised Malaysian man who received a $218 trillion dollar phone bill in 2006, (http://www.msnbc.msn.com/id/12247590 /ns/world_news-weird_news/t/think-your-phone-bill-high-try-trillion/, accessed 30-Sep-2011). These are not serious in scientific computing either as they are readily identifiable as spurious. Such a result is extremely unlikely to mislead.
- Subtly incorrect results. These are results which are profoundly dangerous in scientific computing. It may be thought if they are subtle, they cannot mislead but this was profoundly disproved by [7] who showed that the presence of previously undetected software faults in a comparison of 9 seismic data processing packages independently developed to the same mathematical specifications effectively destroyed the level of agreement to the point where the data became highly misleading. This is exemplified in Figure 1. Although some 17 years old, the programming language used is still in widespread use (Fortran), the packages are still in widespread use, the software processes used are effectively unchanged today and scientific programmers still make mistakes, so its relevance is undiminished.

The experiment of [7] is known as an N-version experiment. Although independent versions of software written to the same specifications do not fail independently, [12], [14], they fail sufficiently independently to provide important insight into quantifying the effects of failure.

Another effective way of achieving this is to release all source code so that it can be independently checked. This is very similar to N-version experiments in that source code is subjected to many analysing minds in parallel. In N-version work, this leads to N versions. In open source work, this will usually lead to one version, independently checked by N persons.

There have been many attempts to build models which can predict unreliability based on some statically measurable property of software, for example the cyclomatic complexity, (a graph theoretic measure of decision complexity), first proposed by McCabe [13] but these have not been very successful, [3].

A typical example of a correlation between recorded defects and the cyclomatic number in a study by Hopkins and Hatton [8] illustrates the unfocused relationship between them with no statistically significant patterns.

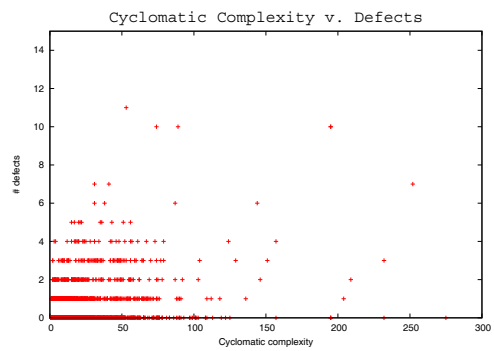


Figure 2: Recorded defects over a period of several years in the NAG Fortran library plotted against the cyclomatic number of the component in which the respective defects were found.

Programming Languages

Problems with ambiguity in programming languages have been reported on many occasions and regularly occur in scientific software in multiple languages, [4]. These include very simple issues like the use of uninitialised variables to far more subtle aspects of computation such as any reliance on a specified order of evaluation, (which is not defined for the majority of programming languages).

Recent research suggests that there are implementation and language independent properties which can be exploited. In [6] I used information theoretic arguments in the

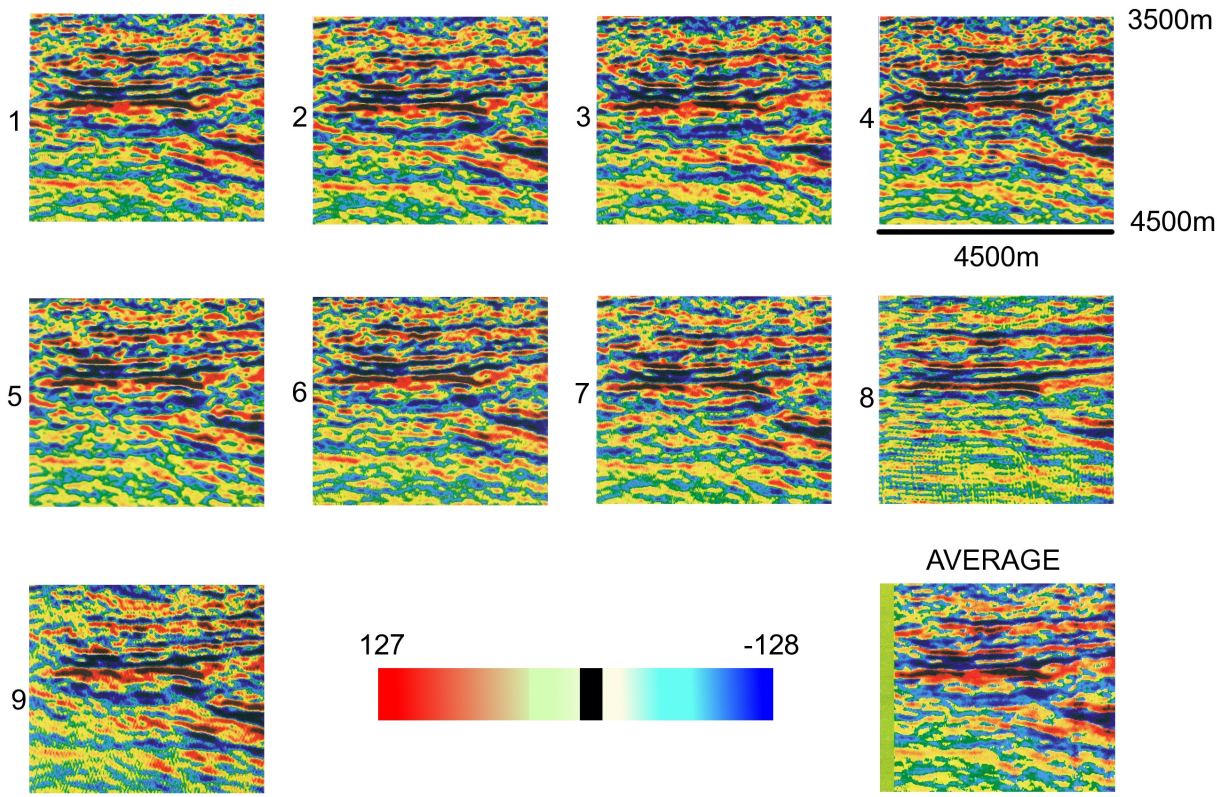


Figure 1: A comparison of nine independently developed packages in the same programming language on the same input seismological data shown by [7]. The y-axis is depth of burial in the earth and the x-axis is distance along the surface of the earth. The outputs vary in the second and sometimes first significant figure. The data needs about three significant figures of accuracy to resolve the geological features (in this case an unconformity trap for a gas field in the North Sea) sufficiently accurately for statistically reliable positioning of a well. The differences were demonstrated to be entirely due to software faults which had been present in the packages in some cases for years. This is by no means unusual as shown by Adams, [1] who demonstrated that a significant number of faults took hundreds and in some cases thousands of execution years to fail for the first time.

context of statistical mechanics to show that the probability p_i of the i^{th} software component in a system containing say, t_i programming tokens in total, obeyed a power-law in the alphabet of *unique* tokens a_i used to build it *independently of the programming language or its application area.* A token here is a keyword, identifier name or operator of a programming language.

$$p_i \approx (a_i)^{-\beta} \tag{1}$$

This in turn leads to a simple prediction that defects will also obey a power-law. One important property of power-law distributions is *clustering* so this gives some theoretical support for the widely-observed phenomenon that defects in software systems do in fact cluster. This can clearly be seen for example in the NAG Fortran scientific library as demonstrated by Hopkins and Hatton [8] as shown in table 1 where all reported defects cluster in only 30% of the executable lines.

Table 1: Defects in NAG Fortran Library

Defects	Components	XLOC
0	2865	179947
1	530	47669
2	129	14963
3	82	13220
4	31	5084
5	10	1195
6	4	1153
7	3	1025
> 7	5	1867

As I pointed out in [6] however, this appears to be a statistical phenomenon with little to be gained by asking the question why are almost 80% of all components defect free. This is akin to asking why somebody has won the lottery, the answer being of course that everybody else didn't.

Clustering is of real value in software testing because if a defect is found, then there is an increased probability of finding another if the same component is re-examined. In other words test effectiveness is improved.

Software Process

Software process in its many forms has become synonymous with High-Integrity developments thanks to initiatives such as the CMMi (Capability Maturity Model Integration) at the Software Engineering Institute of Carnegie-Mellon. This in turn was based on the pioneering work of Humphrey, [9], [10]. The principle notions of this maturity process enabling an organisation to progress through various levels towards full defect management, tracking and prevention have proven to be of great importance.

As with all good things though, there is no question that some implementations of these ideals have become poisoned by bureaucracy and it is all too easy for complacency to creep in, [5]. Nowhere is this more tragically seen than in the events leading up the Afghanistan crash of a RAF Nimrod in 2006, (http://www.official-documents.gov.uk/document/hc0809/hc10/1025/1025.pdf, accessed 01-Oct-2011). In his summing-up, Charles Haddon-Cave QC, author of the independent review stated:

> "Unfortunately, the Nimrod Safety Case was a lamentable job from start to finish. It was riddled with errors. It missed the key dangers. Its production is a story of incompetence, complacency and cynicism."

As important as management of software process is, it should be noted that it makes no guarantees and can shed little light on the effect that residual delivered defects can have on the output of a computation.

Specific Issues in High-Integrity

In the preparation to this paper, I was made aware of a number of issues of relevance in the sphere of Accelerator and Large Experimental Physics Control Systems and I will list these here and make comments on them based on the background above.

Risk versus ingenuity The LHC (Large Hadron Collider) is novel and ingenuity figures large even to get the machine realised. However, risk has always accompanied ingenuity. An ingenious solution is almost by definition new. However new solutions carry risk far more than well-tried solutions because not all of their potential side-effects have been realised. As I stated at the beginning, in terms of dealing with potential safety, this remains a well-rehearsed process with a number of techniques available such as FMECA (Failure Modes, Effects and Criticality Analysis).

Ingenious software solutions are a little different. Ingenuity in software can and often has been associated with obscurity. Obscurity in software is usually a menace because it reduces the power of peer review. Peer review is one of the main weapons in the battle against software defect.

Knowledge transfer and Education This is a vital area. Many of the ways that the problems described earlier in this paper manifest themselves, would be ameliorated with education. Problems with programming languages; the acceptance of basic training in software engineering methods for scientists; engineering technologies such as software redundancy; the imperative for reproducibility; fundamental notions of how to test software and the realisation that software testing is Poppperian in the deniable sense I defined at the beginning of this article, and so on.

In terms of the accuracy of the results of a computation, it is not only the specification, often resident in just a few minds, which is of value. Ultimately, the results depend on the code. The code must therefore be disseminated with the specification to maximise the chance of reproducibility.

At the risk of causing a little upset, I personally have found some scientists highly resistant to the notion that their code is almost inevitably in error. Amongst such scientists, there seems to be the notion that in comparison to scientific research, software implementation is in some sense 'easy'. I have been a practising scientist for much of my career and am a competent mathematician but the formidable difficulties presented by trying to verify the results of a scientific computation are enough to beat humility into anyone who realises just how difficult it is.

Consequently, I would like all people who deal with scientific software to respond to being faced with 100,000 lines of code in the same way, *if we've done a really good job, there will only be around 100 defects in this of which a significant number may well undermine the accuracy of the results.* That is a refreshing, pragmatic and above all, justified approach and I sincerely hope that education will enable this.

Approaches to Design It is commonly thought that High-Integrity systems are specified, built, tested and released. This rather comfortable viewpoint is still frequently taught in universities. In my experience, it could not be further from the truth. In contrast, *every* stage of the implementation of a trustworthy system is accompanied by iteration or prototyping to make sure ideas are feasible before cementing them into a system.

As a very simple recommendation, it is not common for software testers to be included at the design stage. They *must* however be included from the earliest days because testability is an excellent brake on vaulting and frequently unimplementable ambition.

Dealing with technological leaps In High-Integrity systems, the ideal case generally is to ignore them on the grounds that they carry too much risk as described in the section above. However, each must be taken on its own merits recognising that some technological leaps carry great benefit. A very simple example arises from the interoperability and openness of the Internet protocols. The Internet was not designed and no doubt if we had tried, we would have failed. However, from these tiny beginnings of portability and openness allowing cooperating but disparate development, the vast complexity and indeed reliability, of the Internet evolved to everybody's benefit.

CONCLUSIONS

In this essay, I have tried to present some background on why it is so difficult to quantify the inevitable inaccuracy in the results of scientific computation and why the notions of Popperian deniability demand that we make more progress in this vital area.

I have pointed out a few areas in which we are making progress, for example by designing N-versions systems but these are very expensive. Ultimately, there seems no alternative than to open all source code to public scrutiny and hope to garner the same benefits that open source has brought for example to the world of operating systems, where the Linux kernel is now one of the most reliable complex pieces of software the human race has ever built.

Perhaps we can do the same with scientific computation.

REFERENCES

[1] E. Adams. Optimising preventive service of software products. *IBM Journal of Research and Development*, 1(28):2–14, 1984.

[2] B. Boehm, H.D. Romback, and M.V. Zelkwitz. *Foundations of empirical software engineering: the legacy of Victor R. Basili*. Springer, 1st edition, 2005. ISBN 3-540-24547-2.

[3] N.E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5):675–689, 1999.

[4] L. Hatton. The T experiments: Errors in scientific software. *IEEE Computational Science and Engineering*, 4(2):27–38, April 1997.

[5] L. Hatton. Bureaucracy, Safety and Software: a potentially lethal cocktail. In C. Dale and T. Anderson, editors, *Making Systems Safer: Proceedings of the 18th Safety-Critical Systems Symposium*, pages p.21–36, London, UK, 2009. Springer.

[6] L. Hatton. Scientific computation and the scientific method: a tentative road map for convergence. In *IFIP / SIAM / NIST Working Conference on Uncertainty Quantification in Scientific Computing*, 2011.

[7] L. Hatton and A. Roberts. How accurate is scientific software ? *IEEE Transactions on Software Engineering*, 20(10), 1994.

[8] T.R. Hopkins and L. Hatton. Defect correlations in a major numerical library. *Submitted for publication*, 2008. Preprint available at http://www.leshatton.org/NAG01_01-08.html.

[9] W. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989. ISBN 0-201-18095-2.

[10] W. Humphrey. *A discipline of software engineering*. Addison-Wesley, 1995. ISBN 0-201-54610-8.

[11] W. Kahan. Desperately needed remedies for the Undebuggability of Large Floating-Point Computations in Science and Engineering. In *IFIP / SIAM / NIST Working Conference on Uncertainty Quantification in Scientific Computing*, 2011.

[12] J.C. Knight and N.G. Leveson. An experimental evaluation of the assumption of independence in multi-version programming. *IEEE Transactions on Software Engineering*, 12(1):96–109, 1986.

[13] T. McCabe. A software complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.

[14] Meine van der Meulen and Miguel A. Revilla. The effectiveness of software diversity in a large population of programs. *IEEE Trans. Software Eng.*, 34(6):753–764, 2008.

FEED-FORWARD IN THE LHC

M. Pereira, X. Buffat, K. Fuchsberger, M. Lamont, G.J. Müller, S. Redaelli
R.J. Steinhagen, J. Wenninger, CERN, Geneva, Switzerland

Abstract

The LHC operational cycle is comprised of several phases such as the ramp, the squeeze and stable beams. During the ramp and squeeze in particular, it has been observed that the behaviour of key LHC beam parameters such as tune, orbit and chromaticity is highly reproducible from fill to fill. To reduce the reliance on the crucial feedback systems, it was decided to perform fill-to-fill feedforward corrections. The LHC feed-forward application was developed to ease the introduction of corrections to the operational settings. The LHC Feed-Forward software has been used during LHC commissioning and tune and orbit corrections during ramp and squeeze have been successfully applied. As a result, the required real-time corrections for the above parameters have been reduced to a minimum. In parallel, successful trials have been made to apply feedforward corrections before commissioning with beam which are based on MAD-X simulation scans over the unused setting functions. In this paper we present the evolution of feedforward for the LHC and discuss further improvements of this software.

INTRODUCTION

The performance of the LHC is highly dependent on the capability to keep under control some key beam parameters (tune, chromaticity, orbit, ...). These parameters must be kept within acceptable limits to allow safe and efficient operation of the accelerator. Early conclusions pointed out that this challenging task would require the implementation of a real-time control system for satisfactory operation of the LHC. Such system, also known as feedback, would correct perturbing effects on the beam reducing beam loss inside the accelerator. However, the prevailing idea was that the mitigation of external effects on the beam should be done by both feedback and feedforward control [1]. The feedforward control would use the past experience from previous fills and apply corrections to the subsequent ones, eliminating some of the observed effects. The feedback systems would then correct the remaining effects through real-time corrections.

Feedforward is seen as a succession of corrections being applied which become progressively smaller, provided a certain degree of reproducibility exists. Consequently, the measurements obtained will gradually tend to the expected result. In the LHC, feedforward corrections are very similar. Corrections are retrieved from measurements of previous fills and then applied to settings to be used in subsequent fills. Currently, both feedback and feedforward control have been successfuly implemented ensuring safe and performant operation of the LHC. This paper presents implementation details of the LHC Feedforward application, results obtained and discusses future improvements.

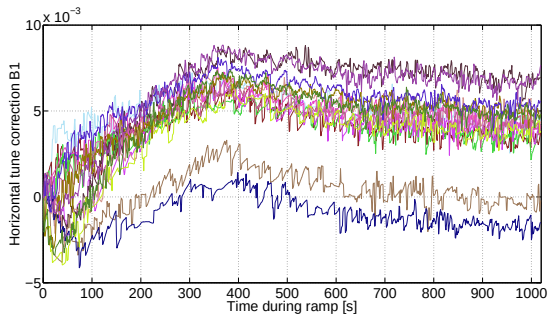


Figure 1: Feedback correction on beam 1 tune on the horizontal plane over 15 fills.

FEEDFORWARD IMPLEMENTATION

Feedback System

Real-time corrections are typically based on a feedback loop. In the case of the LHC one single feedback controller is responsible for correcting tune, chromaticity, orbit, coupling and energy [2]. The hardware receives measurements from the beam position monitors and tune diagnostics systems and calculates the corrections. These are then sent to corrector circuits and RF systems which should stabilize the beam around the established reference value for the beam parameter being corrected. The input data and corrections of the feedback controller are then re-published to be logged in the Logging Database or to be reused in other applications.

Reproducibility

The success of the feedforward correction approach is highly dependent on the reproducibility of the machine on a fill to fill basis. In the case of the LHC, the high level of reproducibility was evident early in beam commissioning. The reproducible behaviour was first observed in tune measurements during ramp. Through the analysis of successive tune corrections made by the feedback systems it was evident that the discrepancy between measurements was quite small. Figure 1 compares feedback corrections of beam 1 tune in the horizontal plane over 15 fills.

The high reproducibility of the LHC came as a pleasant surprise to those who before had seen quite the opposite in LEP. The lack of reproducibility in the machine was evi-

dent and only after 8 years of operation the first feedforward corrections were made.

Logging Database

The Logging Service [3] is responsible for logging all major activity related to the accelerator. This includes a wide range of heterogeneous data ranging from cryogenic temperatures and magnetic field strengths to beam positions and intensities. Data queries are performed either through a dedicated Java API or a desktop tool called *TIMBER*. While the former allows other software applications to access data programatically, the latter is aimed at users who want to explore and visualize the data or eventually export the data to a file.

Smoothing & Sampling

Real-time corrections stored in the Logging Database cannot be directly introduced into LSA. The feedback system logs data at a frequency of 1Hz which translates into a large number of data points that LSA simply does not support. Moreover, the data contains a reasonable amount of noise, which if not removed, would most certainly produce erroneous trim functions. Smoothing is a well known method used in signal processing for mitigation of noise and capture of patterns in data.

There is a wide variety of algorithms but the most commonly used is the ”moving average”. Two variants have been tested during the development of feedforward software: simple or unweighted moving average (*SMA*) and the weighted moving average (*WMA*). The *SMA* is in essence an average value based on a window calculated with the arithmetic mean of subsequences of *n* terms. This average is calculated over an equal number of data points on either side of a central value. Consider the following example of a horizontal tune correction with 10 data points ($P_0, P_1 ... P_9$) and a 5 point averaging window. The points in the extremities are kept intact in the resulting function. The points P_1 and P_8 become the average of their value and of their adjacent points (P_0, P_2 and P_8, P_9 repectively). For all other points the formula is:

$$SMA = \frac{P_{m-2} + P_{m-1} + P_m + P_{m+1} + P_{m+2}}{5}$$

After smoothing, the correction function goes through a sampling process. As the power converters in the LHC cannot ”follow” setting functions for current with data point spacing inferior to 0.1s, caution is needed. Bad data point positioning typically causes a power converter to trip. In the present configuration the LHC Feedforward application will smooth the trim correction using an unweighted moving average of 5-point sliding window and sampling every 10 seconds. This has been proved to be more than enough to efficiently remove signal noise and provide enough data points for setting corrections.

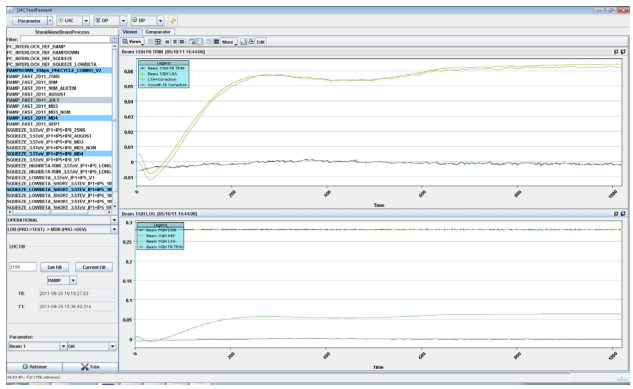


Figure 2: LHC Feedforward application.

LHC Software Architecture

The LHC Software Architecture (LSA) is a framework implemented in Java that covers all essential aspects of control of the LHC such as generation, modification and management of settings, measurements and hardware exploitation. This and other functionalities are made available through a set of clean and well defined APIs divided by domains (optics, parameters, contexts, etc). LSA revolves around three key concepts: *parameter*, *context* and setting. Parameters are organized in a hierarchical structure with each level being dependant on the level above. Thus, changes made on top level parameters are propagated down the hierarchy until the lowest level. The parameters on the top level typically represent physics-related concepts namely tune, chromaticity or orbit, while the lower are related to hardware i.e. power converters. Operators and experts commonly only manipulate top level parameters known as *knobs*. LSA automatically reflects those changes to the lower-level parameters using algorithms called *make rules*. Parameters can have different values depending on the time period considered. This relationship between a parameter value and the time period is called a *context*. There are three types of contexts, *supercycle*, *cycle* and *beam process*. For the LHC, a *context* is called *beam process* and several have been created for the different machine phases: injection, ramp, squeeze, etc.

LHC Feedforward Software

As aforementioned, the LHC Feedforward application has been developed to ease the computation of trim corrections from measurements and their merge with operational settings. It is a standalone application and it is presently available in the CERN control room. The application has been developed using the standard set of technologies for LHC related software: Java, Spring [4] and Swing. Figure 2 shows the interface of the LHC Feedforward application.

At the moment the application is able to perform feedforward corrections on tune, chromaticity and orbit for ramp and squeeze beam processes.

Feedforward corrections on a particular parameter con-

sist of a simple three step process enumerated:

- Logged feedback corrections for the said parameter are retrieved from the Logging Database through its Java API.
- Measurements undergo a smoothing process to eliminate possible noise. Only a subset of these data points are used for the feedforward correction.
- Corrections are applied to the operational settings of the selected parameter using the LSA API.

From the implementation perspective the application is divided into three logical components. The DAO layer is responsible for the communication with both Logging and LSA databases. Direct database access is not possible for security reasons and therefore queries must be done through the already mentioned Java APIs. The business layer contains the correctors and the smoothing algorithms. There are three *correctors*, each one responsible for the calculation of a correction function of a particular parameter, tune, chromaticity or orbit. From the Java implementation perspective these correctors are in separate classes implementing a common Java interface. Hence, *correctors* are homogeneous in functionality and the implementation of a new corrector for a new beam parameter simply requires the implementation of the corrector Java interface. The user interface was created in Swing reusing some of the available standard components from LSA. The interface is divided in three subpanels: a panel for the selection of the beam process to which corrections will be applied; a panel for the selection of the parameter to be corrected and the fill from which the feedback corrections will be taken; a visualization panel where current parameter settings, the proposed correction and the corrected settings are displayed. Additionally, a rudimentary mechanism for comparison of settings and measurements has been implemented. Although quite limited in functionality it allows some brief analysis of fill and settings data.

SIMULATION BASED FEEDFORWARD

In addition to the feedforward of corrections calculated from logged feedback correction data, a second path was used to reduce the stress on the feedback systems. For the optimization of the duration of the nominal squeeze [5] tools have been developed in the toolchain for online modeling [6] that allow to transfer the power converter (PC) *K* parameter settings to MAD-X and recalculate the optics functions. After the settings are generated in a beamprocess for the LHC the online model framework application, `Beamprocess Scanner` is used to calculate the optics functions from the settings extracted at a given time in the PC setting functions. Among other features, the evolution of the optics key parameters (tune, chromaticity,...) can be plotted and stored. The value functions are inverted and applied as correction trims to the available highlevel parameter knobs.

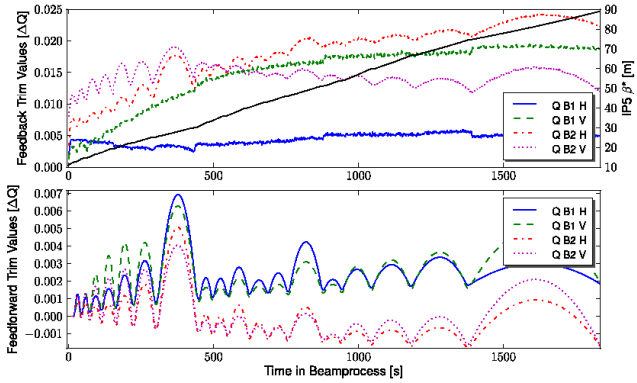


Figure 3: Evolution of the tune knob trims along the 90m un-squeeze. The upper plot shows the real time trims of the tune feedback applied during the un-squeeze, the lower the calculated feedforward corrections from simulation (only the beam 1 corrections were applied). The β* in IP5 from 11 to 90m is shown in the upper plot as well.

This procedure was successfully applied for the tune of beam 1 during the commisioning of the 90m Un-Squeeze [7]. Figure 3 shows the comparison of the tune correction trims required by the feedback system and the proposed corrections from the simulation run. The profit of the feedforward can be clearly observed by the nearly flat real-time trims for beam 1 in the horizontal plane. Real-time trims very similar to the proposed feedforward correction were required for beam 2 where no feedforward had been applied.

RESULTS

The first real test of feedforward corrections was during the commissioning of the first 1.18 TeV ramp. The first ramp trial had already failed due to major tune shifts and the tune feedback was not operational. These were the perfect testing conditions to prove the worth of feedforward corrections. Despite the software being immature and the algorithms not optimized, the accelerator successfully completed the ramp. Since then corrections for chromaticity and orbit have been commissioned for ramp and squeeze. The development focused first on the ramp process which was the more problematic. The next beam parameter to be corrected was chromaticity. Ever since the initial commissioning only one feedforward correction was needed until the present day. Finally, orbit trims have been performed both in ramp and squeeze. Support for orbit corrections was more complex due to the number of parameters involved. Corrections are introduced in LSA at the *K* level since no *knobs* have been defined. Corrections have been successful although very infrequent.

Figure 4 shows the reduction in tune feedback correction after feedforward corrections had been applied to ramp settings.

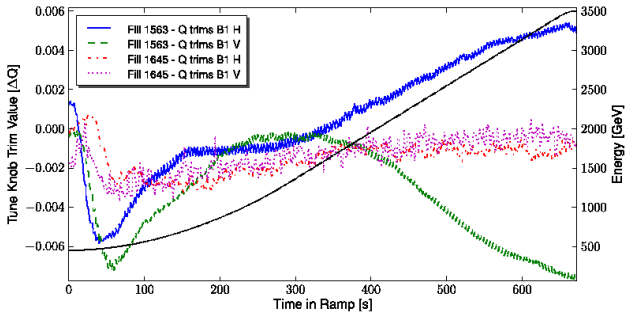


Figure 4: Comparison of the evolution of the tune feedback trims for two fills: 1563 before feedforward and 1645 after feedforward.

CONCLUSIONS & PROPOSED ENHANCEMENTS

The LHC Feedforward application can currently perform corrections during the ramp and squeeze for tune, chromaticity and orbit. The software has been successfully used during 2010 and 2011 achieving the expected results. In parallel, the real-time feedback system has been performing well and was used during every ramp and squeeze in the past year. Hence, feedforward corrections haven't been pursued rigorously and are only performed when new beam processes are created. In addition, due to the high reproducibility of the machine very few corrections were needed throughout 2010 and 2011. However, one should not underestimate the importance of feedforward corrections since they ensure that the beam is not lost in case of feedback malfunctions has occasionally occurred. The simplicity of the LHC Feedforward from an implementation standpoint should be credited not only to the remarkable reproducibility of the LHC but also on the exceptional capabilities of the LSA system. One of the major shortcomings of the application is its limited capability for comparison of parameter corrections and settings over the course of time. This feature would facilitate analysis the effect of feedforward corrections over several fills.

Morever, the application requires a manual selection of the beam process to be corrected, an action prone to mistakes. At the moment of this writing a new feature has been added to the LSA API which allows the retrieval of the name of the beam process used during a determined time period. The integration of this functionality in the LHC Feedforward application would automatize the selection of the beam process and lessen the possibility of mistakes.

REFERENCES

[1] T. Wijnands et al., *Requirements for Real Time Correction of Decay and Snapback in the LHC Superconducting Magnets*, Proceedings of EPAC'00, 2000.

[2] R.J. Steinhagen et al., *Commissioning and Initial Performance of the LHC Beam-based Feedback Systems*, Proceedings of IPAC'10, 2010.

[3] C. Roderick, *The LHC Logging Service - Capturing, storing and using time-series data for the world's largest scientific instrument*, Proceedings of UKOUG'06, 2006.

[4] Spring Source, http://www.springsource.org/

[5] X. Buffat, G. Müller, et al., *Simulation of linear beam parameters to minimize the duration of the squeeze at the LHC*, Proceedings of IPAC'11, 2011.

[6] G. Müller, S. Redaelli, et al., *Toolchain for online modeling of the LHC*, these proceedings.

[7] S. Cavalier, G. Müller, et al., *90m Optics Commissioning*, Proceedings of IPAC'11, 2011.

ATLAS ONLINE DETERMINATION AND FEEDBACK OF THE LHC BEAM PARAMETERS

J. Cogan*, R. Bartoldus, E. Strauss SLAC, Menlo Park USA
D. Miller, University of Chicago, Chicago USA
On Behalf of the ATLAS Collaboration

Abstract

The High Level Trigger of the ATLAS experiment relies on the precise knowledge of the position, size and orientation of the luminous region produced by the LHC. Moreover, these parameters change significantly even during a single data taking run. We present the challenges, solutions and results for the online luminous region (beam spot) determination, and its monitoring and feedback system in ATLAS. The massively parallel calculation is performed on the trigger farm, where individual processors execute a dedicated algorithm that reconstructs event vertices from the proton-proton collision tracks seen in the silicon trackers. Monitoring histograms from all the cores are sampled and aggregated across the farm every 60 seconds. We describe the process by which a standalone application fetches and fits these distributions, extracting the parameters in real time. When the difference between the nominal and measured beam spot values satisfies threshold conditions, the parameters are published to close the feedback loop. To achieve sharp time boundaries across the event stream that is triggered at rates of several kHz, a special datagram is injected into the event path via the Central Trigger Processor that signals the pending update to the trigger nodes. Finally, we describe the efficient near-simultaneous database access through a proxy fan-out tree, which allows thousands of nodes to fetch the same set of values in a fraction of a second.

INTRODUCTION

ATLAS is one of several large experiments situated on the Large Hadron Collider (LHC) at CERN [1]. To cope with the LHC's extremely high bunch crossing rate, currently ≈ 20 MHz, ATLAS makes real time decisions to accept or reject event data using a trigger system [2]. Several algorithms in the software trigger, such as vertex finding and b-jet identification, rely on specific knowledge of the luminous region to maintain a high identification efficiency for interesting physics events.

However, the parameters of the luminous region–the beam spot–change significantly during the course of an LHC fill. The width typically grows $\approx 3\ \mu m$ while the position can drift by $5-10\ \mu m$. To operate at maximum trigger efficiency, we measure and communicate the beam spot parameters to the trigger farm in near real time. We require over 100,000 vertices to measure the 1,300 colliding bunches in the current LHC structure. Running a vertexing algorithm on the trigger farm itself provides access to the large statistics needed at a high rate. Since each trigger process operates independently of the others, our challenge becomes coordinating them to measure the beam spot and then feed it back to each of them for use in tracking algorithms. This measurement and feedback is accomplished without disrupting data taking.

* jcogan@cern.ch

EXPERIMENTAL SETUP

While ATLAS is comprised of several sub detectors, we will focus on the silicon trackers and the trigger system. We use these systems to observe and reconstruct charged particle track and vertices.

ATLAS Silicon Trackers

The Pixel detector, comprised of three concentric cylindrical layers and three end-cap discs on each side, has an excellent hit resolution of $\sigma_{r\phi} \approx 10\ \mu m$ in the plane transverse to the beam axis, and $\sigma_z \approx 115\,\mu m$ in the direction of the beams. The Semi-Conductor Tracker (SCT), surrounds the Pixel detector and has four concentric cylindrical layers in the central region, as well as nine disks on each side. The SCT's silicon is shaped in strips and provides $\sigma_{r\phi} \approx 17\mu m$ and $\sigma_z \approx 580\mu m$ resolutions. The trigger system uses data from both detectors for tracking and vertexing, however the final vertex resolution is dominated by the Pixel hits. They cover an acceptance range of $|\eta| < 2.5$ in pseudo-rapidity, or about 10 degrees from the beam axis.

ATLAS Trigger

ATLAS employs a three level trigger system, one in hardware (L1) and two in software called the High Level Trigger (HLT). A detailed description can be found elsewhere. Here we restrict ourselves to the points related to the beam parameter determination.

Central Trigger Processor For our purposes, the L1 Central Trigger Processor (CTP) performs two important roles. First, the CTP tells the ATLAS Data Acquisition System what is the current LumiBlock (LB). A LB is a period of data taking with nearly equivalent detector conditions, often lasting 60 seconds. It is important that all the events in the same LB use the same beam spot for consistency checking. Second, the CTP has a small data fragment it adds to the data stream as the events are read into the HLT. This fragment contains information about the L1

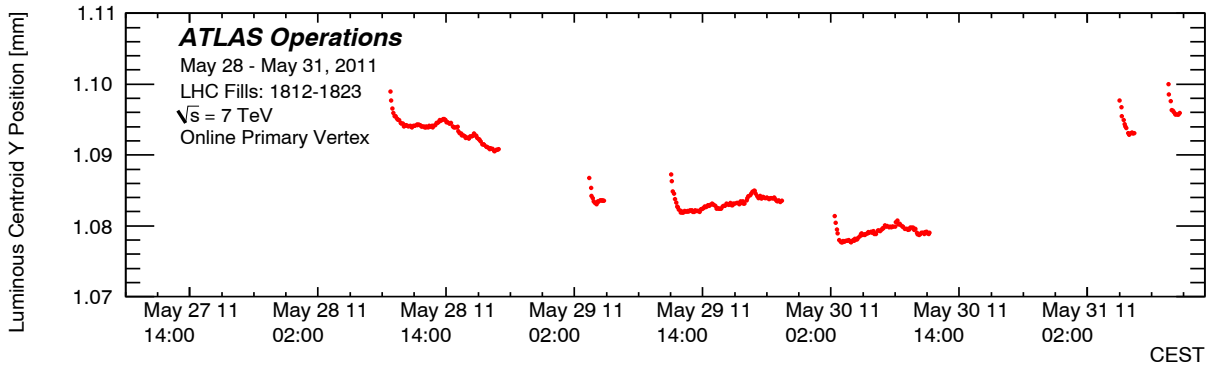


Figure 1: Time-variation of the luminous centroid position in y measured in the High Level Trigger every five minutes, for six separate LHC fills recorded over the span of four days.

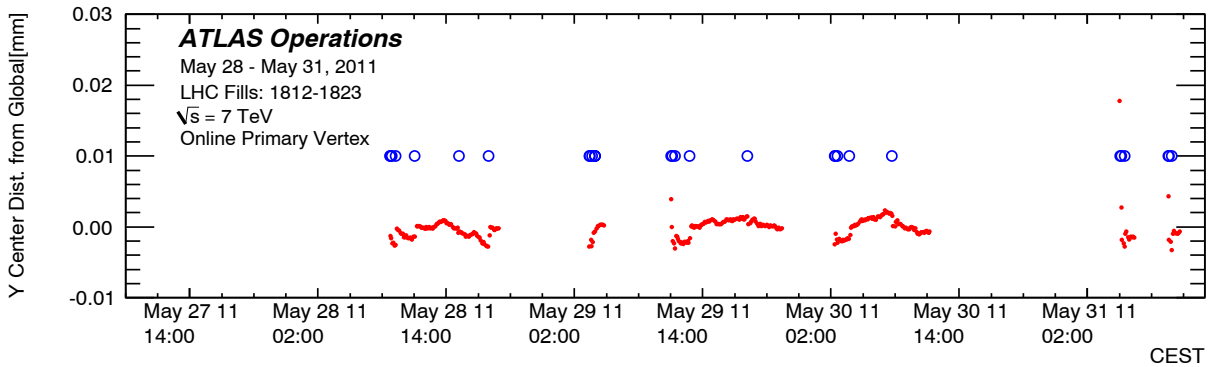


Figure 2: Time-evolution of the deviation of the luminous region position in y. The deviation represents the difference between the most up-to-date measurement and the value stored in the conditions database. The blue circles mark the time of the updates while the red dots show the deviation in millimeters as a function of time. Each red data point uses five minutes of data calculated on the High Level Trigger Farm, for six separate LHC fills recorded over the span of four days.

trigger decisions, the LB, and which set of beam spot parameters to use.

ATLAS High Level Trigger The ATLAS HLT comprises two software triggers: Level-2 (L2) and the Event Filter (EF). The L2 and EF sequentially decide to accept or reject events based on physics objects they have reconstructed. The HLT is run on commercially available server racks with more than 6,000 L2 and EF processors each.

The L2 trigger is the first level to access tracking information and is able to process approximately 75,000 events per second, only examining each event's respective Region of Interest. Due to the presence of tracking information and the high attainable rate of events, the beam spot algorithm runs on the L2 farm for all events that passed certain L1 multi-jet triggers.

The Beam Spot Algorithm

Events which pass the L1 trigger are considered by the beam spot algorithm if they contain sufficiently high quality vertices. When the algorithm processes an event, it calculates the position in x, y, z of the three vertices in the event with the highest number of tracks. Each of these positions is added to local histograms and is used to compute the center of the beam spot and its width (still convolved with the detector-resolution). To calculate the width of the beam spot we employ a split vertex method (described elsewhere [3]), whose primitives are also histogrammed.

METHODS

While building and maintaining this system presents many challenges in tracking and commissioning, we will focus on the hurdles related to the coordination and cooperation of the High Level Trigger Farm.

As we discuss the various components of the feedback system we will mention the latency each one introduces. We define latency to be the time between an event passing the L1 trigger and our first ability to execute an update based on that information. The total latency is approximately 5 minutes.

Gathering

As the beam spot algorithm runs, it accumulates statistics in the local histograms. First we must aggregate these histograms along with those from other applications running concurrently, across all 6,000 L2 processors. We accomplish this via a *Gatherer* tree, in which the leaf nodes

are the trigger processors. Every 40 seconds, the $\approx$ 240 trigger processors in the same server rack send their histograms to the rack level gatherers which sums and stores them locally. At the next 40 second interval the $\approx$ 30 rack level gatherer send their histograms to the top level gatherer where they are summed to produce farm level histograms. From here the farm level histograms are pushed to a historical archive, where asynchronous monitoring applications existing off the L2 or EF farms can query them. To mitigate possible out-of-phase problems between the gatherer and the beam spot monitoring tool, we wait another 60 seconds, bringing the total latency at this step to 140 seconds.

Calculation

Once the histograms are summed across the L2 farm, they are analyzed to extract the beam spot parameters. This requires fitting Gaussian functions to the vertex position histograms to find the distribution mean and width, which correspond to the beam spot center and raw width respectively. The split vertex method is used to calculate the ATLAS vertexing resolution which is then deconvolved to produce a corrected width. Here we calculate both the beam-average parameters as well as per bunch measurements.

We log the calculated parameters, regardless of whether an HLT beam spot update was executed. The parameters are published to the ATLAS online conditions database (COOL) and the LHC logging database (TIMBER). The complete calculation and logging adds 15 seconds to the update latency.

Update Criteria

We do not update the HLT with each new set of beam spot parameters. While the system could handle the continuous feedback, it complicates the following data analysis and adds little extra precision. Thus, there are four criteria, any of which can trigger an update. Each criteria is based on the *current* and *nominal* beams spot. The *current* beam spot refers to the set of parameters just calculated from the HLT histograms. The *nominal* beam spot, on the other hand, refers to the beam spot parameters in use by the HLT. These criteria are completely configurable and were selected to optimize the performance of the displaced vertex triggers. They are as follows:

- The position difference (current to nominal) is greater than 10% of the width in any direction.
- The width difference is greater than 10% the nominal width in any direction.
- The statistical error on a parameter drops by 50%.
- The current beam spot has a valid status flag, while the nominal has an invalid flag. This flag is used to invalidate the beam spot after a beam dump.

The Central Trigger Processor

To update the parameters used by the HLT, the monitoring tool must first indicate to the 13,000 trigger processes that their beam spot is out-of-date. Furthermore, it is imperative that at any given LB the entire trigger farm is using the same set of beam spot parameters.

To accomplish the update, the monitoring application publishes the new parameters to a temporary network location and informs the CTP of the new beam spot.

The CTP writes the beam spot parameters into the conditions database with an interval of validity starting at the next LB, N. At the same time it updates its data fragment with the LB number of the next beam spot update. When the first event in LB N arrives the HLT nodes discover there is a new beam spot available from the information in the CTP fragment. Each trigger process then reloads the parameters from the conditions database. In this way, over 13,000 nodes (L2 and EF) are told there is a new beam spot, with a clean LB transition.

Since the CTP waits for the next LB this step adds 60 seconds to the update latency.

Proxy Tree

There is a final problem: how can 13,000 processes query the conditions database simultaneously? We have two advantages in this situation. Firstly, each process is asking the same question, "what is the beam spot for LB N?". Secondly, each process receives the CTP fragment (and thus queries the database) with some random time offset, $\approx 50\ ms$ relative to the other processes. This offset comes from the average event processing time.

To exploit these advantages we employ a database proxy tree built on the CORAL abstraction layer. Instead of directly querying the conditions database, the trigger process asks a database proxy which may or may not already have the response cached locally. If a proxy does not have the response it passes the request further up the tree. If the top level proxy does not know, it asks the CORAL server which queries the Oracle conditions database. As the response goes back down through the tree, each proxy stores the query-response pair locally and thus prevents unnecessary database access in the future.

Thus, only a few trigger processes need to wait the full round trip time to query the database; most nodes wait only for their nearest proxy to retrieve the answer from local memory. On average, each process only waits $\approx 10\ \mu s$. As the waiting time is short compared to the average processing time at L2, this induces no dead time on the HLT. Finally, the top level servers, CORAL and Oracle, are saved from attempting to field 13,000 requests.

RESULTS

The automatic beam spot determination and feedback has been in place since early 2011. In the following section we summarize its feedback performance and some of the physics results produced from its logging output.

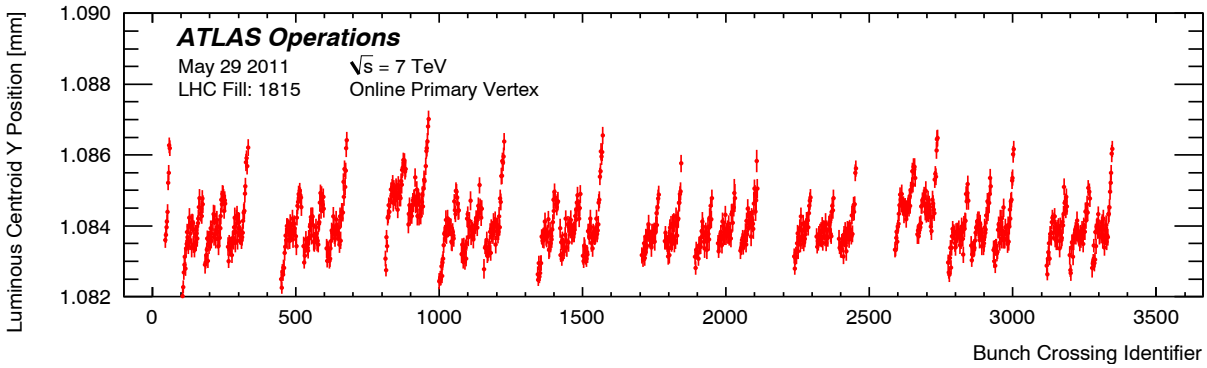


Figure 3: The luminous centroid position in y measured in the High Level Trigger for each of the 1024 colliding bunch pairs separately. Distinct structures are visible, with variations of up to 5 μm and repeating patterns across the injected bunch trains.

Feedback

Here we discuss four measures of the feedback mechanism: latency, frequency, system load, precision.

- Latency: As discussed in **Methods** section, the update mechanism has a minimum amount of time between an event being observed and an update it might trigger. The total latency is $\approx$ 4 minutes which is shorter than typical beam behavior and satisfies the trigger needs.
- Frequency: At the beginning of a fill the update mechanism first fires five minutes after collisions begin (because of latency) and then 4-5 times in the first 30 minutes due to the rapid decrease of statistical errors. After this early phase, beam orbit drift and emittance blow up are the most common causes of beam spot updates. These updates happen much less frequently, typically occurring only once every few hours.
- System Load: Once the HLT nodes receive the new CTP fragment, they must all fetch the new beam spot. However due to the proxy tree this causes only a 10 ms pause for each process on average. This load is so small that it induces no system dead time.
- Precision: As discussed in **Update Criteria** section the feedback mechanism maintains the difference between current and nominal beam spot parameters small by occasionally updating the HLT. In Figures 1 and 2, we see the current and nominal beam spots are kept within a few microns despite 30 micron variations in the current value over the course of a week.

Physics

The occasional updates of the beam spot used by the HLT keeps tracking-triggers operating at a high efficiency. In addition to the updates, the monitoring tools also record the parameters across time and for each colliding bunch separately. We provide time lines of the position and width in all three directions, and the tilt with respect to the beam axis of the luminous ellipsoid. Furthermore each value is available for offline analysis and stored with the statistical error from the fitting process.

The online system is uniquely positioned to perform the 1,300 per bunch calculations because of the high rate of good vertices available at L2. Capitalizing on the high rate of events, we calculate the center of the luminous region for each bunch to within $\approx 1\mu m$, which is better spatial resolution than the LHC dedicated hardware can obtain. Examining the bunch position variation within a train, shown in Fig. 3, we see unambiguous evidence for beam-beam kicks from long range collisions at the LHC. Long range collisions occur when a bunch in beam 1 passes near, but not through, a bunch in beam 2. Each bunch exerts an electromagnetic force on the other distorting its orbit and shape.

CONCLUSION

We have developed a system to calculate and monitor the LHC beam parameters using the ATLAS High Level Trigger Farm. Furthermore, if the nominal beam spot parameters used by the HLT tracking algorithms differ significantly from the current values an update is executed while the system continues to collect data.

We coordinate the communication to and from the 13,000 independent HLT processes by exploiting the gatherer and proxy systems already in place. Finally, the HLT's beam spot is maintained within microns of the current value while inducing no dead time on the data taking system.

REFERENCES

[1] ATLAS Collaboration "The ATLAS Experiment at the CERN Large Hadron Collider," *JINST*, vol. 3, p. S08003, 2008.

[2] ATLAS Collaboration "Expected Performance of the ATLAS Experiment - Detector, Trigger and Physics," no. CERN-OPEN-2008-020, 2009. http://cdsweb.cern.ch/record/1125884

[3] ATLAS Collaboration "Characterization of Interaction-Point Beam Parameters Using the pp Event-Vertex Distribution Reconstructed in the ATLAS Detector at the LHC," ATLAS-CONF-2010-027 http://cdsweb.cern.ch/record/1277659/

BEAM BASED FEEDBACK FOR THE LINAC COHERENT LIGHT SOURCE*

D. Fairley, K.Kim, K. Luchini, P. Natampalli, L. Piccoli, D. Rogind, T. Straumann
SLAC National Accelerator Laboratory, Menlo Park, California, U.S.A.

Abstract

Beam-based feedback control loops are required by the Linac Coherent Light Source (LCLS) program in order to provide fast, single-pulse stabilization of beam parameters. Eight transverse feedback loops, a 6x6 longitudinal feedback loop, and a loop to maintain the electron bunch charge have been commissioned on the LCLS, and have been maintaining stability of the LCLS electron beam at beam rates up to 120Hz. This paper will discuss the design, configuration and commissioning of the beam-based Fast Feedback System for LCLS. Topics include algorithms for 120Hz feedback, multicast network performance, actuator and sensor performance for single-pulse control and sensor read back, and feedback configuration and runtime control.

INTRODUCTION

The Linac Coherent Light Source at SLAC requires several beam-based feedback systems to stabilize electron beam parameters. In April 2010 the LCLS began operating at a 120Hz beam rate. Several architectural changes were made to the LCLS control system in order to stabilize the beam at 120Hz. These changes were implemented in two phases over the last two years, and finalized this summer with the successful commissioning of the LCLS Fast Feedback system. This paper discusses the architectural changes required to achieve single-pulse stabilization at the 120Hz beam rate, the commissioning process and the performance of the feedback system.

FEEDBACK REQUIREMENTS OVERVIEW

The LCLS feedback systems stabilize several beam parameters through feedbacks of three basic types, transverse feedback loops, a longitudinal feedback loop, and a few simple single parameter general feedback loops. These feedback loops were prototyped in MATLAB and had been maintaining stability of the LCLS beam until the summer of 2011.

Transverse Trajectory Feedbacks

Transverse trajectory feedbacks have been commissioned at nine locations along the linac where the launch into the downstream section is important. They stabilize the X and Y position and angle of the beam. The majority of them run at 10Hz rep-rate. The Linac-to-Undulator (LTU) Feedback operates at beam rate, up to 120Hz, for single-pulse stabilization of the beam position and angle before entering the undulator region. See Figure 1.

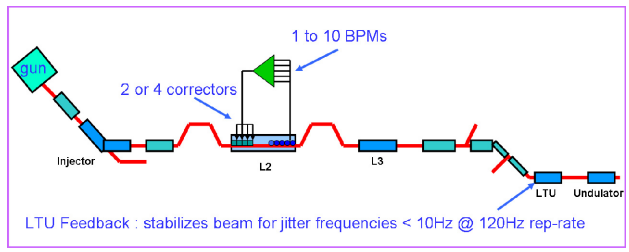


Figure 1: Transverse Feedbacks.

Longitudinal Energy and Bunchlength Feedback

The longitudinal feedback is responsible for single-pulse stabilization of the beam energy at four locations along the LCLS linac, and the bunch length at two locations along the linac. This feedback maintains all six parameters simultaneously using a 6x6 matrix of transfer coefficients [1]. Figure 2 is a schematic of the longitudinal energy (δ) and bunchlength (σ) feedback. The longitudinal feedback operates at beam rate, up to 120Hz.

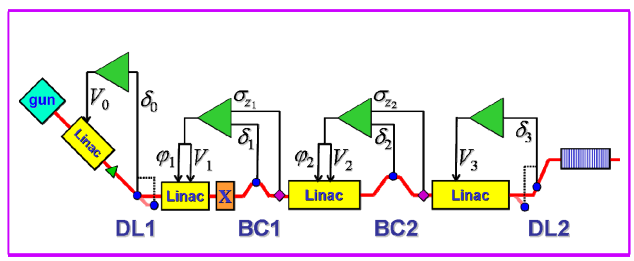


Figure 2: Longitudinal Feedback [2].

Other Simple Feedbacks

The Fast Feedback system allows for addition of feedbacks through its modular 'framework' design. The feedback engineer can define a feedback by entering a set of sensors, actuators, an algorithm, and identifying the parameters to maintain. Several simple single sensor, single actuator feedbacks were planned for the LCLS. The Bunch Charge feedback is an example; it maintains the charge by adjusting the laser intensity based on the charge reading of a beam position monitor.

120 Hz Feedback

In April 2010 the LCLS began operating at 120Hz beam rate. For 120Hz feedback the time-critical devices

*Work supported by U. S. DOE Contract DE-AC02-76SF00515

are the corrector magnets, which require 6ms to settle to within 85% of the reference point. Figure 3 is a diagram of the message timeline required to correct the 120Hz beam. This diagram shows that the feedback calculations must be completed within 2ms to successfully make corrections with the magnets. We further divided the 2ms, giving the sensors 1ms to produce measurements, and the feedback processors 1ms to calculate and send corrections to the actuators. [3]

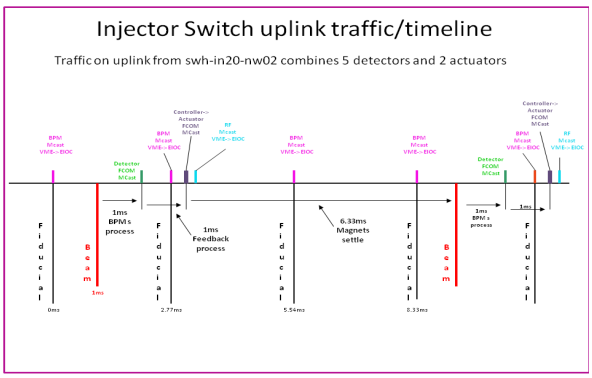


Figure 3: LCLS Fast Feedback message timing diagram.

120 Hz beam operation draws power from two interleaved 60 Hz power line phases which have differing noise characteristics that must be corrected independently by the feedback. The LCLS timing system assigns a 'pattern' for each pulse that indicates from which 60Hz "timeslot" the pulse is generated. This pattern-based timing is quite complex and can indicate other properties on each timeslot, such as whether there is a pulse, or the beam-rate.

The feedback processors must interface to the timing system in order to properly correct for each 60Hz timeslot variation. Since these differing noise characteristics include a large DC offset it was further required that the RF and magnet actuators interface to the timing system so that they can manage these power differences as 'offsets' even when the feedback is off.

Additional Requirements

The LCLS has additional features that prove challenging for continuously running feedback loops. Figure 4 shows where the final energy can be set automatically to a value anywhere between 4.3 GeV and 14GeV. The linac configuration can change using bend magnets to direct the beam into dumps at various locations along the linac, or to move the chicanes. A kicker magnet upstream of the undulator region can kick out pulses on a pulses-by-pulse basis, and is used to protect the undulator magnets. The beam charge can be changed from as low as 20pC up to 250pC

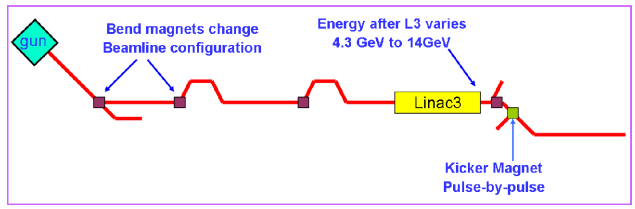


Figure 4: LCLS Changing beam line.

LCLS FEEDBACK COMMISSIONING

In order to meet the 120Hz single-pulse stabilization requirement, and correct for the 60Hz differences the final production feedback system required several architectural changes and additions to the LCLS. These upgrades are described in detail in [4]. The significant changes include:

- Faster network communications
- Timing pattern based processing
- Faster feedback processing

The system upgrades were first developed in a test environment that included a dedicated network with three switches and a router, a timing system processor, two sensor devices and their control processors, a magnet subsystem, an RF control subsystem, and a feedback processor. See Figure 5.

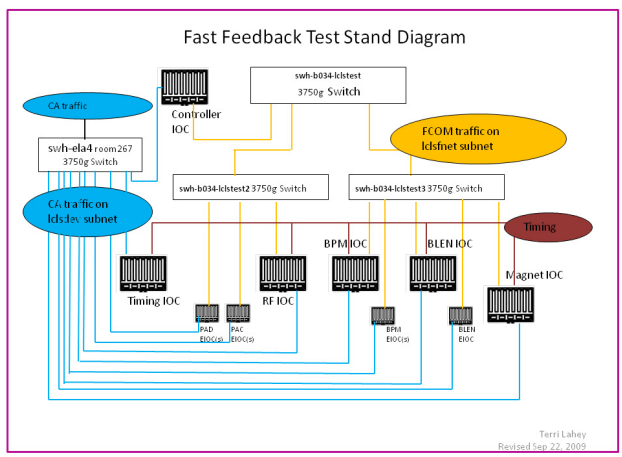


Figure 5: LCLS Fast Feedback test stand.

This full test setup was instrumental to developing these significant changes on schedule. The individual subsystems were debugged, the network was developed and characterized, the configuration tool and EDM displays were developed, and much of the feedback timing and control between subsystems could be worked out without requiring time on the production system. The following sections detail the architectural changes made to the LCLS and describes the performance improvements of the system.

120HZ PHASE I

The 120 Hz Fast Feedback system was installed and commissioned on the LCLS in two phases. Phase I included installing the dedicated feedback network, and

adding a new timing system interface to the actuator devices. During phase I we were allotted three 8 hour shifts with beam to commission the network, sensors and actuators.

Dedicated Network

A dedicated gigabit Ethernet network was added to the LCLS infrastructure, to isolate the feedback communications from all other network traffic. The feedback IOC applications communicate over this new network using an IP-Multicast/UDP based protocol called FCOM, developed at SLAC [5], which works similar to a reflective memory communication scheme. The network interface software provides a simple API and has been developed as a software module that can be added to any IOC that must become part of the LCLS Feedback system. Figure 6 shows the added network. Each IOC used in feedback communicates on this network via a second on-board NIC and the FCOM software module.

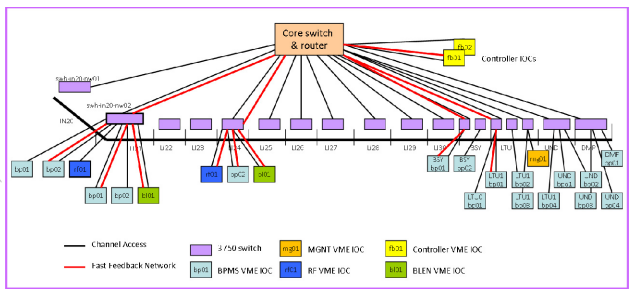


Figure 6: An isolated Feedback network added to the LCLS infrastructure, shown in red.

The Feedback Network is defined as a separate VLAN and includes its own uplinks from the LCLS sector switches to the main switch and router. Test-stand testing demonstrated the feedback communications capable of transporting round-trip messages from a sensor IOC to the feedback controller IOC and back in under 200us.

Timing System Interface

The timing system interface for actuators and feedback loops includes an Event Receiver and interface software module developed at SLAC called a Pattern-aware Unit (PAU) [6]. The PAU is a 'multiplexer' software module, which allows the actuator or feedback IOC to accept set point values per timing pattern. This software module registers with the timing system to receive a fiducial event. It queries the timing system for the current pattern pipeline and chooses the correct set point based on the current pattern. See the diagram in Figure 7.

The PAU allows any LCLS actuator device to discover the 'timeslot' pattern assigned to each pulse in order to synchronize its processing with the pulse.

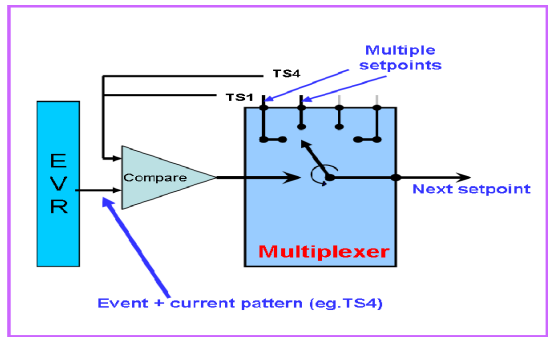


Figure 7: Each PAU input has an assigned timing pattern and actuator set point.

Sensor Devices

LCLS sensors already had an EVR and were interfaced to the timing system. The beam position monitors (BPMs) and bunch length monitors, (BLENs) were made available to the Fast Feedback by integrating the BPM and BLEN control processors with the FCOM software module. On the FCOM network the BPM measurements take less than one msec. to arrive at the feedback IOCs after a pulse, meeting the 1ms requirement.

The BLENs were not able to meet this requirement, but the flexible feedback configuration options allowed us to adjust the longitudinal feedback to delay processing and make up the difference by sending corrections to the RF actuators later in the timeline (the RF system does not require settling time as the magnets do).

Actuator Devices

LCLS RF stations and corrector magnets were integrated with the feedback network with the FCOM module, and the timing system using the PAU module. The new PAU software module proved very successful in aiding beam stability for users before the Fast Feedbacks were commissioned in 2011, and seamlessly handled the addition of the FACET program, which drew additional power on one of the 60Hz power line phases, creating a 30Hz disturbance. Thus the PAU allowed the magnets and RF to correct with an offset at 60Hz and two offsets at 30Hz.

120HZ PHASE II

The second phase of the Fast Feedback commissioning was accomplished in 2011. During this run the Fast Feedback controllers were installed and 11 feedbacks were commissioned using one to four hours of machine development time once per week over several months.

Feedback Controller and Software Framework

In order to gain processing speed, as well as interface to the timing system and the new network, the feedback loop calculations were implemented in EPICS IOCs. The LCLS IOCs consist of a VME6100 processor with two NICs, an EVR, and run the RTEMS real-time operating system.

The Feedback Loop Controller IOC application uses EPICS records for data and configuration storage, and is a multi-threaded application using the EPICS OSI library. This Feedback Controller application can run one or more feedback loops. The Controller IOC application is developed as a software 'framework' that handles common functions, such as managing operational limits on sensor readings, actuator set points, feedback calculated parameters, managing changing machine conditions, and error checking and reporting

Four Controller IOCs were installed and commissioned. The first transverse feedback loop was commissioned over two test days, and then five more were quickly added. The configuration tool described in [4] made it very simple to configure and install the remaining feedbacks, since the framework had already been commissioned. The 120Hz LTU feedback was the last transverse feedback commissioned. The transverse feedbacks average a processing time of less than 400us, meeting the overall 1ms requirement. Figure 8 shows the plots of the working 120 Hz LTU feedback.

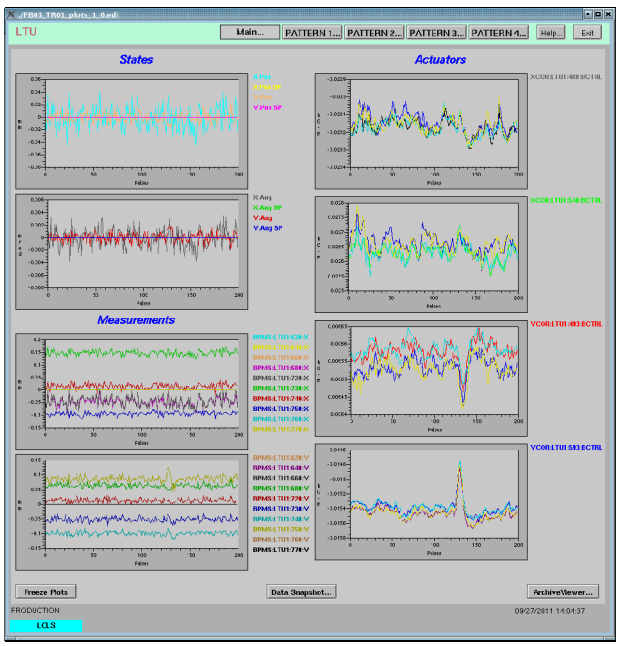


Figure 8: LTU Feedback plots.

The Longitudinal Feedback was the biggest commissioning challenge. As a global feedback, it must maintain energy stability of the beam through the many transitions of the LCLS machine, such as energy changes, charge changes and beam tune-up for each new user. This feedback had to be integrated and tested with many other machine tuning high-level applications and in many different machine configurations.

SUMMARY

Commissioning a feedback system on a production machine is a complex, difficult task. The demands on a working accelerator leave little time for global system commissioning. It is often useful to review the project as a whole and identify efficiencies in hindsight:

- Prototyping the feedbacks in MATLAB allowed machine operation with a fairly stable beam for users until Fast Feedbacks were fully commissioned.
- Investing in an extensive test stand for development allowed engineers to develop enhancements to a near-complete point.
- A 'virtual machine' in the test stand might have reduced even further our need for production test time.
- The network upgrade was inexpensive as it used mainly existing network infrastructure, but it cannot support 360Hz beam rate. A reflective memory solution might have been better for future growth.

Most importantly, commissioning in two phases gave us a fast network and timeslot aware actuators to use at 120Hz before full feedback installation. This allowed us to 'shake out' the bugs for half of this complex upgrade before moving on to the full feedback installation, and later allowed us to stabilize beam adequately during production shifts while commissioning feedbacks on development days.

REFERENCES

[1] P. Krejcik, "Controls Requirements for LCLS Feedback Systems", LCLS Physics Requirements Document #1.1-304. August 2005.

[2] J. Wu et. al., "Linac Coherent Light Source Longitudinal Feedback Model", 2005 Particle Accelerator Conference, Knoxville, TN. USA, May 2005.

[3] T. Straumann, "FCOM Testing in B34(*draft*)", October 2009.

[4] D. Fairley et. al., "Beam-based Feedback for the Linac Coherent Light Source", October 2009

[5] T. Straumann, "LCLS Fast Feedback Communication Infrastructure Interface Control Document", July 2009.

[6] K. Kim, "Development of the Pattern-aware Unit (PAU) for the LCLS Beam-based Feedback System", October 2011.

COMMISSIONING OF THE FERMI@Elettra FAST TRAJECTORY FEEDBACK*

G. Gaio, M. Lonza, R. Passuello, L. Pivetta, G. Strangolino, Sincrotrone Trieste, Trieste, Italy

Abstract

FERMI@Elettra is a new 4th-generation light source based on a single pass Free Electron Laser (FEL). In order to ensure the feasibility of the free electron lasing and the quality of the produced photon beam, a high degree of stability is required for the main parameters of the electron beam. For this reason a flexible real-time feedback framework integrated in the control system has been developed. The first implemented bunch-by-bunch feedback loop controls the beam trajectory. The measurements of the beam position and the corrector magnet settings are synchronized to the 50 Hz linac repetition rate by means of the real-time framework. The feedback system implementation, the control algorithms and preliminary close loop results are presented.

INTRODUCTION

Since the beginning of the FERMI@Elettra commissioning [1], a framework based on Matlab allowed the flexible implementation of slow feedback loops by arbitrarily choosing sensors and actuators from a pool of devices [2]. Based on the response matrix concept and its inversion through the Singular Value Decomposition (SVD), the framework was used to implement a slow trajectory feedback whose main goal was to steer the beam and restore a given trajectory. Since the feedback was a loop written in Matlab running with a period of a few seconds, less importance was given to its dynamic response and ability to damp noise up to a certain bandwidth.

The tight specifications in terms of stability required by a new concept machine such as a seeded FEL together with the uncertainty about the tools necessary to support a challenging commissioning, led to the implementation of a fast feedback system joining the flexibility of the Matlab framework with the capability to interact with the beam shot by shot.

To achieve such results, beam diagnostics (electron and photon Beam Position Monitors – BPMs, charge monitors, bunch length monitors, fluorescent screens, etc.) and actuators (magnet power supplies, RF plants, etc.) communicate in real-time at the linac repetition rate. A distributed shared memory (Network Reflective Memory, NRM) based on a dedicated Gigabit Ethernet network allows deterministic data exchange through the control system [3].

IMPLEMENTATION

Although a feedback loop running at 50 Hz can be easily managed by an up-to-date computer, its components must be carefully evaluated to prevent weird behaviour due to unpredictable execution time. In fact, in order to optimize the feedback performance, the whole process of collecting BPM data, performing feedback calculation and setting correctors, must be carried out in one intra-shot period of 20 ms, thus reserving just a few milliseconds for each of these operations.

CPU

The trajectory feedback relies on the control system front-end computers which interface BPMs and corrector power supplies. An additional computer, called “real-time server”, is dedicated to the feedback processing. They all are VME systems with MVME7100 PowerPC CPU boards running Linux and the Adeos/Xenomai real-time extension. The availability of four Gigabit Ethernet ports onboard allows an easy and effective separation between real-time and non real-time traffic.

The adoption of the Adeos/Xenomai real-time extension reduces the latency and the jitter of critical tasks to the microsecond range. To take full advantage of the system real-time capabilities, all the software which implements data processing and communication with the hardware is executed in Adeos/Xenomai interrupt handlers or tasks. Shared memories and FIFOs allow communication between real-time tasks and Tango servers running in user space.

Sensors

55 stripline (linac) and 25 cavity (undulator region) BPMs provide beam position measurements with a measured resolution of 10 and 4 μm respectively.

The Libera Single-Pass detector, manifactured by Instrumentation Technologies, has been chosen to acquire the bunch by bunch analog signals from the stripline BPMs. The detector features two Ethernet links:

- One 100 Mbit/s port directly connected to the control system network mainly used by the Tango device server, running embedded, to configure the detector parameters (gain, offsets, …) and monitor the electronics status.
- One 1 Gbit/s port providing low latency data by transmitting, synchronously to the machine trigger,

* This work was supported in part by the Italian Ministry of University and Research under grants FIRB-RBAP045JF2 and FIRB-RBAP06AWK3.

UDP packets containing horizontal and vertical beam positions and the four raw stripline waveforms.

The real-time acquisition is done by six VME based control system computers, each handling a group of up to 15 BPMs. A standard Gigabit switch serializes the Ethernet packets coming from a group of detectors and sends them to the CPU.

On the CPU side, a dedicated Ethernet port, which is in charge of collecting data in real-time, uses a customised driver to provide raw access to its interrupt routine. Once the packets are received, a kernel module application executed in the interrupt handler extracts and copies beam position information in the NRM and in a local circular buffer.

In the worst case, the whole trajectory is acquired by the front-end computers in about 140 μs.

The cavity BPM detector consists of two homemade MicroTCA cards, ADO and ADA [4]. ADO acquires analog signals directly from the cavity BPM front-end and transmits the data using a UDP packet. ADA drives the BPM calibration signal which is generated after each shot. Both boards provide a 100 Mbit/s Ethernet port that will soon be upgraded to 1 Gbit/s.

The cavity BPMs acquisition system is similar to that of the stripline BPM, except that the position calculation and the detector supervision are directly managed by the low level computers. Three control system computers are reserved for data collection through dedicated Ethernet switches.

Actuators

The trajectory correction is done by two types of magnets: air-cored, used in the low energy part of the linaca nd in the undulator region, and iron-cored, used in the rest of the machine. Two homemade magnet power supplies (model A2605 and A2620 respectively) provide 5 or 20 Amps. The external interface relies on a 100 Mbit/s Ethernet link and a simple UDP protocol allowing to drive the output current and get the status of the power supply at a maximum rate of 200Hz. Eleven low level computers, installed along the accelerator, interface the power supplies through dedicated Ethernet switches.

A kernel module application running in each computer receives synchronization and correction data from the NRM. A timer of the CPU working at 10 KHz, started every machine shot, sequentially set the power supplies connected to it. At each timer interrupt, the kernel module routine sends an UDP packet containing the current setting to one power supply; the timer is stopped immediately after the last power supply is set. The generated current is the sum of the setting provided by the Tango server and the one from the trajectory feedback. A single low level computer manage up to 40 power supplies.

Real-time Communication

The NRM plays a fundamental role interconnecting in a deterministic way all the 20 CPUs involved in the trajectory feedback. It also provides the "bunch number", a sort of "real-time timestamp" which univocally identifies each linac shot.

At present, the total amount of data exchanged on the NRM by the trajectory feedback is about 3 KB per shot. It mainly contains BPM data, corrector values and the related bunch number. The maximum propagation time across the NRM is 1 ms.

The feedback acts on the electron beam bunch by bunch, which means that the correction of a given bunch is calculated based on the measured position of the previous bunches. To maximize the dynamic performance of the feedback, the open loop total delay must be at most one shot period, thus 20 ms. The following is the time schedule of the whole feedback chain within one shot period starting from the generation of a given bunch number:

- **0 ms**: time zero, bunch number starts propagation on the NRM.
- **1 ms**: new bunch number available on the NRM.
- **1.2 ms**: linac shot, the BPMs acquire the beam position.
- **2.2 ms**: new beam positions available on the NRM.
- **3 ms**: feedback routine calculates corrector values.
- **7 ms**: new corrector values available on the NRM.
- **9 ms**: all the corrector power supplies are set with the new values.

The remaining time is sufficient for the magnet current to reach the final value and to provide the required magnetic field to the incoming bunch.

Feedback Processing

Although, for practical reasons, the trajectory feedback routines run on one dedicated CPU, the flexibility of the system allows running bunch by bunch feedbacks on every CPU which joins the NRM.

Similarly to the rest of the real-time software, the feedback code is written as a Xenomai kernel module. In particular, to take advantage of the floating point operations, which are not supported in the standard Linux kernel, the algorithm code runs in a Xenomai thread.

A Tango device server, interfaced to the kernel module by a shared memory, configures at runtime the list of sensors and actuators, sets the control parameters and loads or measures the response matrix for each of the feedback loops. Each Tango device server implements about 100 Tango commands/attributes.

The control algorithm is based on a standard PID controller which is preceded by either a configurable low pass or median filter; the latter has better performance in presence of spiky data. A series of notch filters could be configured to remove efficiently periodic noise sources. As the whole feedback chain is carried out in the time period within two shots, the closed loop dynamics is dominated by one period delay.

The response matrix is empirically calculated by measuring the trajectory perturbation produced by each corrector. The measurement procedure is integrated in

the feedback software and is carried out with a bunch by bunch measurement. The power supplies are driven sequentially with a programmable current ramp. In the meanwhile the BPMs are acquired synchronously to the corrector excitation. At the end of the process, an algorithm calculates the response matrix correlating ramped kicks and trajectory distortions. To halve the measurement time, the process can also be performed in parallel on both planes.

The SVD algorithm is employed to calculate the correction matrix. In the "inversion" process, correctors, singular values and BPMs can be individually weighted. The correction efficiency and robustness to mesurements errors can be optimized by means of singular value reduction with the algorithm known as Tikhonov regularization.

COMMISIONING

During the FERMI@Elettra commissioning no relevant periodic noise has been measured on the beam, except for a ripple introduced by a bending magnet power supply, which has been eventually replaced. Temporary malfunctions of the timing distribution and the low level RF systems have erratically affected the beam trajectory, but also these problems have been solved by fixing the source of the disturbance. Slow drifts, which are mainly related to temperature variations and slow changes in the characteristics of the accelerating cavities, are presently the main noise sources that the feedback is supposed to counteract (Fig. 1).

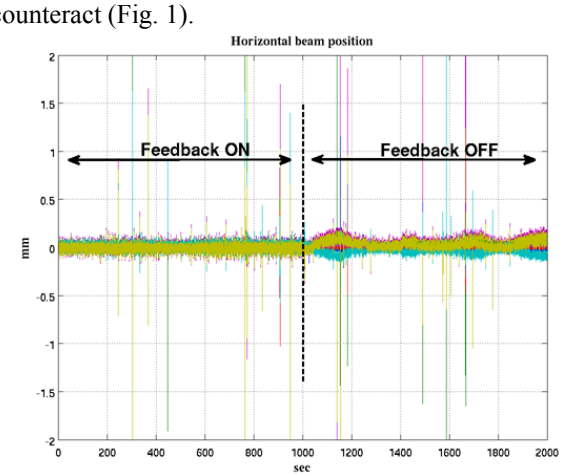


Figure 1: Feedback effect on the horizontal trajectory measured with 34 BPMs in the linac.

Micro interruptions of the electron beam due to quick trips of the RF plants and lasting a couple of linac shots, are automatically recognized by the software using simple heuristic rules and filtered out.

In order to study the closed loop behaviour, a Matlab simulation code has been developed. It allows the rapid tuning of the control parameters by manually optimizing the system response. The sensitivity function calculated with the simulation using a given set of PID parameters (kp=0.05, ki=0.2, kd=0) agrees quite well with the one measured on the real machine (Fig. 2).

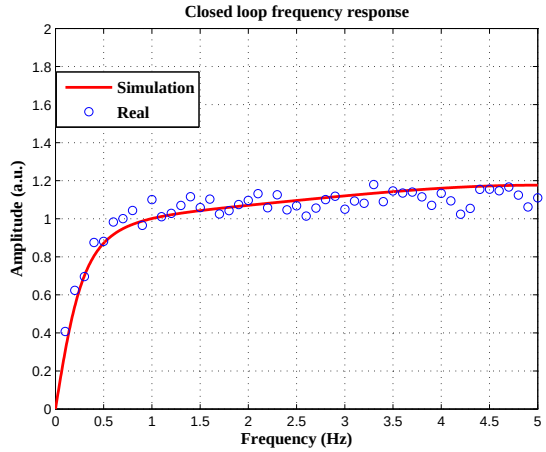


Figure 2: Simulated vs. real sensitivity functions with the linac working at 10 Hz repetition rate.

At present two fast trajectory feedback loops are used during commissioning shifts (Fig. 3). The first controls the trajectory in the linac by means of 34 BPMs and 34 correctors. The main goal is to keep stable the beam trajectory in the accelerating structures. The second makes use of 8 BPMs and 8 correctors, and act on the beam position in the undulators. With the present configuration of the control parameters no evidence of disruptive coupling between the two loops has been noticed. Nevertheless a global approach using all of the BPMs and correctors in a single loop will be pursued in the next future.

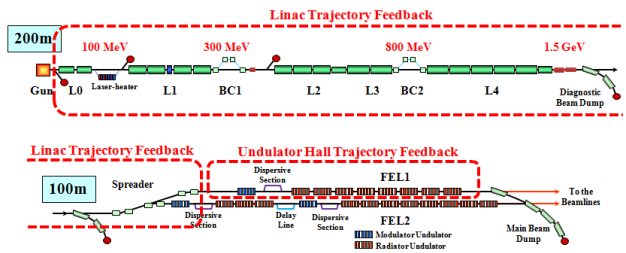


Figure 3: The machine sectors where the two feedback loops are currently operating.

The fast response matrix measurement procedure demonstrated to be very efficient. By using a four-shot ramp per corrector, the measurement of the linac response matrix in both planes takes less then 20 seconds (10 Hz repetition rate).

Small variations of the beam energy, changes of the beam transport optics, uncorrelated noise coming from the electronics and non linearity of the BPMs can lead to discrepancies between the response matrix and the real machine, and produce an increase of the beam position *rms* when the feedback is on. The amplification of the noise is concentrated in the higher part of the spectrum. In order to mitigate this problem, the Tikhonov regularization method has been adopted. It allows a good correction at the lower frequencies while the higher part of the spectrum is not amplified significantly. This method has the additional advantage of avoiding the

correction of unreal trajectories that often brings to useless high corrector strengths.

Although the fast feedbacks have been successfully used during the first experiments on the beamlines, the most frequent usage is during the machine commissioning with the following purposes:

- Perform trajectory scans inside the accelerator cavities to find a condition that minimizes the beam emittance.
- Restore a given golden trajectory.
- Keep stable the trajectory when changing the phases of the RF plants.

Each feedback loop can be easily operated and monitored from the control room using a QTango [5] graphical panel (Fig. 4).

The general configuration of the feedback including correction and control parameters can be changed also by non-expert people. In addition to the standard ON/STANDBY/OFF buttons and trajectory plots, a *SlowMode* checkbox reduces automatically the feedback gain by a factor of ten. When the feedback is run just once to restore or steer a trajectory, the *SteerMode* checkbox controlling the Tikhonov regularization factor is enabled, and the singular value reduction is not applied.

A useful graphical feature using slide bars allows selecting just a part of the machine and to close the loop on it. This is accomplished on the fly by setting to zero the weights of BPMs and correctors that must be excluded.

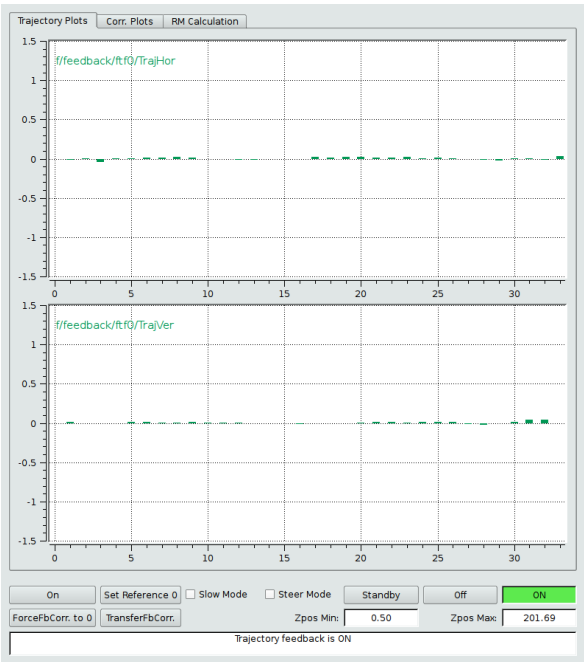


Figure 4: Fast trajectory feedback control panel.

CONCLUSION

The experience gained in developing fast feedback systems for the Elettra storage ring [6] together with the availability of a powerful and well integrated real-time framework, helped the development and commissioning of the fast trajectory feedback, which is now regularly used during machine operations.

Additional features are planned to facilitate its operation and increase the robustness in case of disruptive events such as unexpected changes of the beam energy due to trips of the linac accelerating sections.

REFERENCES

[1] S. Di Mitri, "Commissioning and Initial Operation of FERMI@Elettra", IPAC 2011, San Sebastián, September 2011.

[2] M. Lonza et al. “Beam-Based feedbacks for FERMI@Elettra Free Electron Laser”, IPAC’10, Kyoto, May 2010.

[3] L. Pivetta et al. “The FERMI@Elettra distributed realtime framework”, these proceedings.

[4] A.O. Borga et al., “The MicroTCA Common Acquisition and Processing Back-end System for FERMI@Elettra Diagnostics Instrumentation”, these proceedings.

[5] G. Strangolino et al., “QTango: a Qt Based Framework for Tango Graphical Control Panels”, ICALEPCS’09, Kobe , October 2009.

[6] M. Lonza et al. “A fast orbit feedback for the Elettra storage ring”, ICALEPCS’07, Knoxville, October 2007.

THE DESIGN OF THE ALBA CONTROL SYSTEM: A COST-EFFECTIVE DISTRIBUTED HARDWARE AND SOFTWARE ARCHITECTURE

D. Fernández-Carreiras, D. Beltran*, T. Coutinho, G. Cuní, J. Klora, O. Matilla, R. Montaño, C. Pascual-Izarra, S. Pusó, R. Ranz*, A. Rubio, S. Rubio-Manrique, R. Suñé*,
CELLS, Cerdanyola del Vallès, Barcelona, Spain

Abstract

The control system of Alba [1,2] is highly distributed. The hardware infrastructure for the control system includes in the order of 350 racks, 17000 cables and 6300 equipments. More than 150 diskless industrial computers, distributed in the service area and 30 multicore servers in the data center, manage several thousands of process variables. In this environment, the software client server model, with fast and reliable communications, was imposed. Tango[3] plays an important role. It is a big success story of the Tango Collaboration, where a complete middleware schema is available "off the shelf". Moreover, Tango has been effectively complemented with Sardana SCADAs (Supervision Control And Data Acquisition) [4,5], a great development effort shared and used in several other institutes.

The whole installation has been coordinated from the beginning with a complete cabling and equipment database, where all the equipment, cables, connectors are described and inventoried [6,7]. The cabling database, or "ccdb" can be considered as the core of the installation.

This paper explains the design and the architecture of the control system, describes the tools and justifies the choices made. Finally, it presents and analyzes the figures regarding cost and performances.

INTRODUCTION

Alba is a third generation synchrotron located near Barcelona in Spain. The Installation of the Control System for the Accelerators finished at the end of 2010. The final functional tests took place during the first weeks of 2011, right before the commissioning of the Storage Ring (see Fig. 1), which started the 8th of March of 2011. Currently we are carrying out the final stages of the installation and commissioning of the seven Beamlines of the first phase.

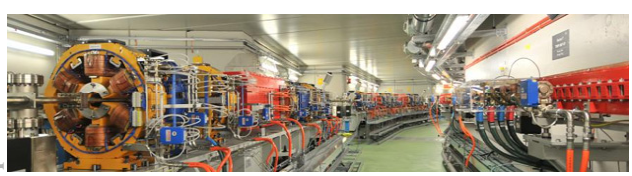


Figure 1: Picture of the Storage Ring (left) and Booster Accelerator (right) in the tunnel.

ARCHITECTURE AND PHILOSOPHY

Typically equipments are located in the tunnel and their controllers and electronics are installed in cabinets in the service area. This results in hardware being distributed all over the place. With this highly distributed installation, we needed as a general rule, distributed software, combined with a fast and robust fieldbus. On the other hand, although we have real time requirements, they are managed by the hardware avoiding the complexity and costs of managing a real time operating system such as RT Linux, VxWorks or RTEMS.

Ethernet as a Fieldbus

Ethernet is the standard for both control networks and fieldbuses. This makes the installation homogeneous, easy to maintain and guarantees a good longevity of the components. The network is physically configured as a star with two main switches having a backplane of 800Gbps each, and 20 Gbps links with the aggregators. It is logically organized in Virtual LANs (VLAN) for separating sectors and subsystems. There are independent VLANs for Safety, Diagnostics CCD cameras, Monitoring and Controls. In addition, every sector has a VLAN for Controls handling Beam Position Monitors, Power supplies, diagnostics, diskless computers, etc. The control room has workstations for running human machine interfaces all connected by Ethernet without any other application specific cable. The only exception is the Personnel Safety System (PSS), which has an independent Safety Bus, and an independent cabinet in the control Room.

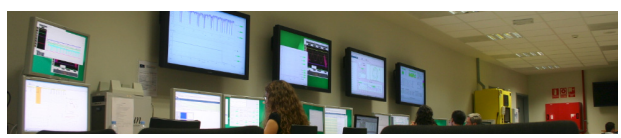


Figure 2: Picture of the Control room for the Accelerators.

The different pieces of software run on diskless compact PCI and industrial PCs, distributed in the service area with direct access to the hardware devices. The boot servers, archiving, Tango databases, CCD data acquisition and various other services, like electronic logbooks are centralized in the computing room (showed in the Fig. 2), where virtualization is broadly employed.

Tango as the Middleware

Tango was chosen among the three options we considered (Tango, EPICS [8], commercial SCADA). Although EPICS had clear advantages in several aspects, like a significantly wider catalogue of tools and a much wider community, Tango was preferred because of its

*On leave

newer technologies, object orientation, and definitely, because most members of the controls team at that time, having worked at the ESRF[9] before, were more familiar and in contact with the Tango community. Choosing Tango, we also disallowed having real-time and deterministic links in our middleware. We use a standard Linux distribution (currently openSUSE 11.1), as the main operating system for controls, with no real time extensions, which makes it cost-effective and easier to deploy and to maintain. Due to few very specific cards, a Windows installation is required in few computers (currently Windows XP), for which drivers were only available for this operating system. This is the case of the Low Level Regulation of the Radiofrequency cavities.

Human Machine Interfaces

Commercial SCADAs, have notoriously evolved over the last fifteen years. After the "crisis of the dot-com" at the beginning of the last decade, many companies were merged or absorbed in bigger ones. Nowadays, the number is still considerable, although, Wanderware and Siemens WinCC are the most popular. They used to be "monolithic" in the past, then they became distributed and today they turned out to be "Networked" in order to match recent requirements. They are integrating upwards with ERPs, and whereas in the past the challenge was to support a large variety of hardware, today the challenge is to ensure consistency of the information, atomic transactions and cyber security. Nevertheless, there are many examples of commercial SCADAs in particle accelerators, like "Global Screen" at Soleil [10] or PVSS (now absorbed by Siemens) at CERN [11] among others. However often in the industry, SCADAs cover the whole manufacturing process, whereas in scientific installations is often limited to human machine interfaces.

Commercial SCADAS did not fulfil our requirements. We needed an optimized access to hardware, a powerful and flexible environment for sequencing and macro execution, software synchronization, generic graphical interfaces, and access to save/restore facilities. For these reasons, we promoted a development of a scientific SCADA on top of Tango: Sardana [4]. Sardana also covers a complete framework for graphical interfaces: Taurus [12].

Real Time Hardware, non Deterministic Software

There are some cases, in which deterministic links are mandatory, such as synchronization or interlocks, where the system cannot afford having sporadic delays or loosing communications. We managed this determinism by suitable hardware, where dedicated systems handle the specific requirements.

The Equipment Protection System [EPS] [13] uses B&R hardware. It comprises about 50 PLCs and more than 100 distributed periphery supervising more than 7000 signals. Temperatures, water flows, valves, flow switches, compressed air, and vacuum valves, gauges and pumps are controlled by the EPS. Equipments inside the tunnel are integrated in shielded boxes with the remote periphery, guaranteeing a better modularity, shorter cables and facilitating the maintenance works.

All CPUs are intercommunicated by an independent deterministic network (Ethernet PowerLink), with a cycle time of 10 ms in most cases, ensuring a response time for interlocks of that order. Every experimental station has an independent EPS, communicating with the machine through hardwired digital inputs and outputs.

The Personnel Safety System is based on Pilz PLCs, SIL3 compatible following the norm IEC 61508. It is totally independent concerning hardware and software and relies on a dedicated safety bus with about 150 ms cycle time.

The timing system [14][15] is based on events and is built with hardware provided by MRF (MicroResearch Finland). It transmits the events using a clock of 125MHz synchronous with the RF frequency and a few picoseconds jitter. Each event receiver is configured to perform an action (i.e. activate pulse on an output) for a particular event code. There is one event generator and about 100 Event Receivers installed on the diskless compact PCI in the service area.

This infrastructure also handles the Fast Interlocks, redundant to the PLCs interlocks. This is an innovative system implemented by MRF upon Alba requirements. The system has a 8 nanosecond resolution providing time stamps and buffering of events, required for post-mortem analysis. In total 418 elements of the accelerator are synchronized and 49 fast interlocks are managed with a response time of about four microseconds.

AUTOMATIC CONFIGURATION AND CODE GENERATION

In order to facilitate the installation, and reduce maintenance time and costs, a central information repository has been setup: the equipment and cable database [6]. It is a relational database implemented on mysql, where types of equipments, cables and connectors are declared allowing later naming and managing instances of every item. This is the repository used for cabling installation, generating labels, cable check and validation. It has been extensively used during the preinstallation, and is constantly updated and kept up-to-date. Figure 3 shows a view of its web interface.

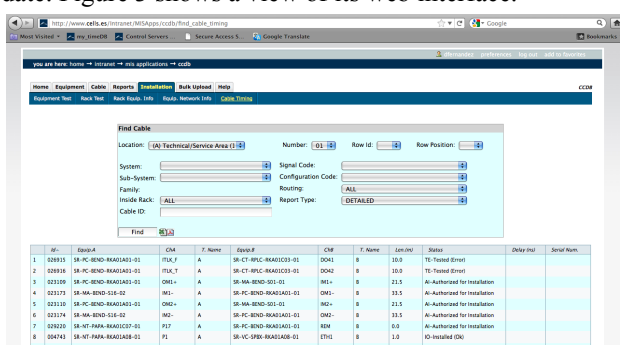


Figure 3: Snapshot of the web interface of the Equipment and Cabling database.

This is critical, since it is used for automatic software generation for PLCs, some Graphical interfaces and several files for configuration of network services, like DHCP, Radius and DNS.

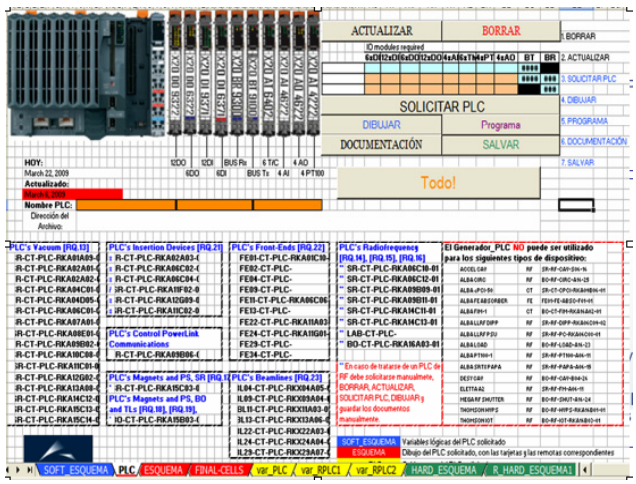


Figure 4: Excel application for EPS code generation.

Many components of programs running in the PLCs of the Equipment Protection System are generated automatically from the database. The huge effort done for keeping the database up-to-date and consistent pays back here, when software components are configured automatically, reducing the errors and increasing the efficiency. A Visual Basic Script running in excel generates another file containing the declaration of variables, data structures, software tasks, modbus mapping and documentation [16]. Figure 4 shows a view of the front page of the excel file. The Logic is still programmed manually due to the difficulty managing the large number of exceptions to the standard interlock rules. In addition, Tango attribute names and the expert GUIs are also generated from the controls equipment and cabling database.

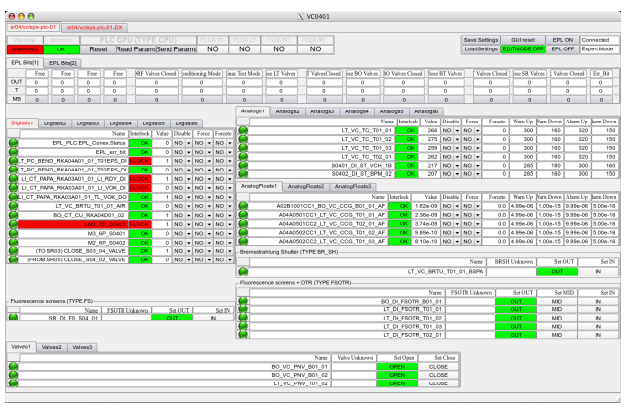


Figure 5: One of the expert GUIs of the EPS.

Important parts of the control system have also been configured from the cabling database. A python API [17] has been created for access to the ccdb, which provides functions for different purposes like extracting all information about equipments and connections between equipments. For example, for each vacuum device (gauge, pump or valve), all ports and the controller corresponding to that device, as well as the serial line and controls computer associated are extracted from the database. Other valuable pieces of information are also queried from the ccdb like the particular PLC which manages a given pressure interlock, the vacuum valves associated to that interlock, etc. Furthermore, the information available in the cabling database is used to extend the displays of the control system and allocate equipments in generic browsers. Figure 5 shows a view of a graphical user interface generated from the ccdb. The cabling API enables applications to sort and search by location of controllers or by location of the gauges connected to them. Also, graphical interfaces benefit from a centralized repository. For example a grid widget may query the cabling database to obtain the name of the instrument connected to a port, and show it as a label. Figure 6 shows the general user interface for vacuum controls, which uses few widgets automatically generated.

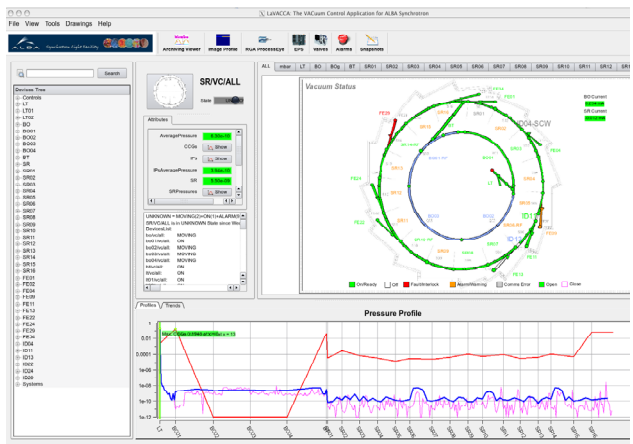


Figure 6: Snapshot of the vacuum control general graphical user interface (Vacca).

COSTS AND BENEFITS

We have considered price and functionality as important constraints for the design of the control systems Tango is cheap in terms of licences. It is distributed under the LGPL conditions and is free of charge. We have a similar case regarding the Operating Systems. Most distributions are free (openSUSE11.1). Windows workstations and servers cover less than 10% of the installation. Most programming is written in Python, for both control and data acquisition.

However, in several cases commercial packages like Matlab are needed. Packages for accelerator modeling like Matlab Middle Layer, need Matlab licences.

The choice of cPCI and Industrial PCs made the bill considerably cheaper compared to VME. We considered savings of at least 30%. ADLINK proved to be a cost-effective solution for cPCI crates, CPUs, data acquisition cards (DAQ) and Industrial PCs. ADCs are used in most places, whereas the use of voltage to frequency converters and counters cards is limited to few cases.

B&R PLCs proved to be a cost-effective solution. They are small powerful and above all, economical if we compare them with the market leaders such Siemens.

Concerning cabling, the fact of having a highly distributed architecture with electronics installed inside shielded boxes located in the tunnel, allowed shorter and cheaper cables. For a machine of 268m, we have an average length of 20 meters (36 if we consider only pure signal cables) with a final average cost of 120 Euros per cable. This is only a very rough estimation since for the price depends much on the type of cable. A high voltage cable for vacuum devices is much more expensive than a connection to a thermocouple, for example.

The having Ethernet as a fieldbus increased the cost of the network installation for the control system, but reduced significantly, the number of serial lines, which installation and commissioning is more time consuming, and we reduced significantly the number of cable types. CCD cameras, oscilloscopes, power supplies etc. are all controlled by Ethernet, reducing drastically the number and the length of coaxial cables.

CONCLUSION

The design of the control system had as main premises being functional and economic. From the beginning, a special attention was paid to cost and effectiveness. Choices such as Ethernet as a fieldbus proved to be correct since now (five years after the design of the control system of Alba) is the common choice. Currently during the commissioning, choices like a compact and economic PLC, or the absence of VME (extremely popular in this type of installations), are working very well. The tendency is to reduce the number of computers and rely even more on Ethernet.

Moreover, in order to make the installation and maintenance easier, a central repository (so called cabling database or "ccdb") was created. It soon turned out into a central repository for the whole installation, kept up-to-date and from where important pieces of software were automatically created. Some examples are the declaration of variables in PLCs, files for configuration of network services, dynamic attributes in device servers and components of graphical interfaces. This is a great tool for the maintenance since the information related to the installation is only in one place.

CONTRIBUTIONS

The whole controls, electronics, network system administration groups actively participated in the design, installation and tests of the control system. Among them, the person managing the installation of the Cabling Toni Camps and Beamline electronics engineering, Julio Lidón had a important role. Other persons which actively contributed are Zbigniew Reszela, Antonio Milán, Maciej Niegowski, Sergi Blanch, Jairo Moldes, Lothar Krause and Fulvio Becheri. Furthermore, the help of the Management Information System group, in particular, Oscar Sánchez, Isidre Costa, Alberto Nardella, Anne-Cecile Klora, Daniel Salvat and Valentí Prat, who developed and maintained the cabling database, the TimeDB and the Project Management system has been crucial. Furthermore we would like to thank the whole the ESRF in particular Emmanuel Taurel, Alejandro Homs, Vicente Rey and Laurent Claustre, who were involved in the early stages, and the other partners in the Tango Community, Nicolas Leclercq, Pascale Betinelli and Alain Buteau, from Soleil, Mark Heron from Diamond, and Timo Korhonen from SLS, who were always available to answer questions and give valuable advice.

REFERENCES

[1] D. Einfeld et al. "ALBA Synchrotron Light Source Commissioning". IPAC 2011. San Sebastian. Spain.
[2] http://www.cells.es. "The synchrotron light source Alba"
[3] http://www.tango-controls.org. The Tango official website.
[4] T. Coutinho et al. "SARDANA: The software for building SCADAS in Scientific Environments". ICALEPCS 2011. WEPMS023
[5] http://www.tango-controls.org/static/sardana/latest /doc/html/index.html The Sardana documentation.
[6] D. Beltran et al. "The Alba controls and cabling database". ICALEPCS 2009. Kobe.
[7] D. Fernández-Carreiras et al. "The commissioning of the control system of accelerators and Beamlines at Alba Synchrotron". ICALEPCS 2011. Grenoble. MOPMU006.
[8] http://www.aps.anl.gov/epics EPICS home page.
[9] http://www.tango-controls.org/static/taurus/latest /doc/html/index.html. The Taurus Documentation.
[10] http://www.esrf.eu. European Synchrotron Radiation Facility.
[11] http://www.synchrotron-soleil.fr. Soleil Synchrotron. The French National Synchrotron.
[12] http://www.cern.ch. European Organization for Nuclear Research.
[13] D. Fernández-Carreiras et al. "Personnel Protection, Equipment Protection and Fast Interlock systems. Three different technologies to provide protection at three different levels". ICALEPCS 2011. Grenoble WEPMU005.
[14] O. Matilla et al. "Alba Timing System. A known architecture with a Fast Interlock System Upgrade". ICALEPCS 2011. Grenoble. WEAAUST01.
[15] J. Moldes et al. "he MRF Timing System. The Complete Control Software Integration in Tango". ICALEPCS 2011. Grenoble, Francia. MOPMU023.
[16] D. Beltran et al. "The Alba controls and cabling database". ICALEPCS 2009. Kobe.
[17] S. Rubio-Manrique et al. "A Bottom-up Approach to Automatically Configured Tango Control Systems". ICALEPCS 2011, Grenoble, Francia. MOPMN003

TOWARDS HIGH PERFORMANCE PROCESSING IN MODERN JAVA BASED CONTROL SYSTEMS

M. Misiowiec, W. Buczak, M. Buttner, CERN, Geneva, Switzerland

Abstract

CERN controls software is often developed on Java foundation. Some systems carry out a combination of data, network and processor intensive tasks within strict time limits. Hence, there is a demand for high performing, quasi real time solutions. Extensive prototyping of the new CERN monitoring and alarm software required us to address such expectations. The system must handle dozens of thousands of data samples every second, along its three tiers, applying complex computations throughout. To accomplish the goal, a deep understanding of multithreading, memory management and interprocess communication was required. There are unexpected traps hidden behind an excessive use of 64 bit memory or severe impact on the processing flow of modern garbage collectors. Tuning JVM configuration significantly affects the execution of the code. Even more important is the amount of threads and the data structures used between them. Accurately dividing work into independent tasks might boost system performance. Thorough profiling with dedicated tools helped understand the bottlenecks and choose algorithmically optimal solutions. Different virtual machines were tested, in a variety of setups and garbage collection options. The overall work provided for discovering actual hard limits of the whole setup. We present this process of architecting a challenging system in view of the characteristics and limitations of the contemporary Java runtime environment.

OVERVIEW

Diagnostics and Monitoring at CERN

Device monitoring and alarm management systems have long been running separately as part of the CERN accelerator controls environment [1]. Both deal with a vast quantity of diverse devices scattered around CERN installations to acquire, process and redistribute their data. What makes a difference is the meaning of data. Monitoring system periodically collects the statuses of the supervised infrastructure. In principle, data shows no irregularities, as the underlying equipment is by and large healthy. The alarm system, however, carries the information already known to have indicated abnormal conditions. They must be swiftly delivered, through the Operators, to the experts capable of solving the problem. In one case both systems clearly intersect: when the monitoring data flags an erroneous situation and must be converted into an alarm.

Prototyping

Recently it has been decided to prepare and evaluate an architecture of the combined solution, a system that would coalesce all the types of data, devices and users, embracing both worlds. Significant effort has been put to model several possible outcomes based on well established CERN software packages [2][3]. To constitute a solid foundation for such analysis and discover genuine physical boundaries we would certainly face, an independent prototype was constructed [4].

For prototyping it was decided to only cover the most critical topics. The outcome was focused and tuned to maximal extent to satisfy the hardest requirements. From a technical viewpoint such effort helped us to:

- determine if the requirements can be realistically matched with available components,
- establish hard limits for performance & scalability,
- find optimal solutions for the most resource-consuming operations.

In this paper we discuss selected subjects - design choices, issues, best practices - that have become important in achieving expected quality.

ARCHITECTURE

Problem Description

CERN monitoring system works in a heterogeneous environment. It has to communicate with a variety of *devices*, through a set of different protocols. Many devices are regular CERN Linux servers. Majority, however, typically front-ends to the physical equipment, operate on less standard or legacy architectures, unknown to Java products. Devices provide monitoring or alarm data in form of *metrics*, sample values describing their state. Metrics are published either periodically (*monitoring agents*) or immediately after an erroneous condition is discovered (*alarm agents*). Devices and the network that connects them are *not fully reliable. No real-time* infrastructure is used to communicate with them. Once delivered to the monitoring server, metrics have to be verified, correlated, converted into alarms, etc. Such an individual *calculation* based on metric(s) is defined in a *rule*. Decisions are made which of them and in what form should be afterwards delivered to the clients - the GUI applications that present the state of the CERN infrastructure to the Operators. Information flow must be *preserved* for future inquiries too.

Requirements

The system is expected to fulfil a long list of requirements, of which a subset relevant to this paper is presented. For this paper we focus on the performance, scalability and reliability issues. The system must be at the time of writing able to:

- acquire alarm metrics from around 10000 devices,
- deliver alarms within bounded time frame,
- acquire 50 metrics/min from 2500 monitoring agents,

- in total, on average, handle 2000-5000 metrics/sec,
- in total, on average, perform around 5000-20000 rule calculations/sec,
- sustain traffic peaks in form of alarm avalanches with several times bigger throughput.

Considering constant expansion of the CERN infrastructure, the amount of equipment is bound to increase. Thus, our system would hopefully accommodate for double the numbers presented above.

Among the points omitted here are: two-way communication with devices, managing misbehaving devices, reconnections, filtering the input, failover mechanism or any configuration aspect.

Design

The overall design follows a classic multilayer architecture with clear separation of concerns. The bottom tier is represented by scattered devices - data providers. Above lies a data acquisition tier (DAQ), which caters for a variety of middleware protocols to communicate with the devices. Those include JMS, Yami4 [5], SNMP and CERN extension of CORBA called CMW [6]. DAQ is located on a separate machine due to the nature of its intensive self-contained computations. Another machine hosts the middle-tier whose main component is a rule engine (RE), a core calculation unit of the system. All the components above the device layer are built in Java.

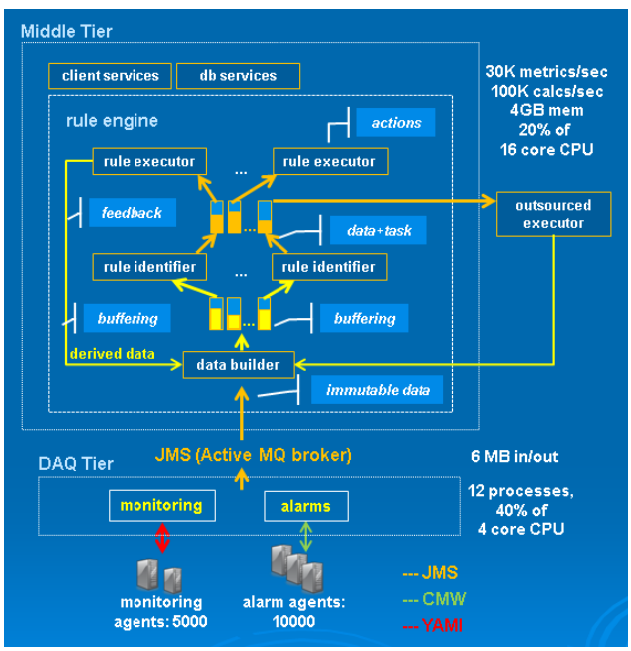


Figure 1: System architecture.

Figure 1 gives a system overview. The numbers associated show the final setup and the results of enduring tests performed with the ultimate architecture. Bottom tier consists of various classes of devices supervised by their DAQ counterparts using middleware protocols.

Now we decompose the presented architecture to look thoroughly at the issues solved throughout and major topics that had arisen.

TECHNICAL PATH

Uniformity

Our bottom tier is a heterogeneous environment. The messages holding data samples are of different shape, the middleware they flow through is divergent. What we pursue however is the opposite: a unification of resources.

CERN control systems largely rely on JAPC library [6]. JAPC operates with different middleware protocols while preserving a common high-level API. It adopts the concept of a *parameter,* which describes a device and a class of data it offers. For each parameter JAPC is able to establish an appropriate transmission channel to its underlying device. JAPC unifies data acquisition across the whole bottom tier. It seamlessly consolidates the access to over 10000 parameters.

The data itself is consistently shaped thanks to introducing a common *metric* structure. It encapsulates every type of input data. JAPC monitoring threads translate the incoming traffic into a stream of unified metrics, whereby acting as pipeline data converters. Such metrics are complete: they hold all the information required for further processing. From then on they are marked immutable, which is crucial to efficient data distribution and task parallelism of the middle-tier.

DAQ outgoing stream of metrics is again divided into new parameters. Such parameters represent logical domains rather than individual devices (alarms vs. monitoring data, computers vs. equipment, etc). It renders an amalgam of almost randomly acquired source data into a narrowly classified stream of metrics. Moreover, DAQ can decide on the pace those metrics are fed into the output JMS channel (recurrent vs. continuous flow) and the size of messages holding them. The latter prevents JMS environment from being flooded with miniscule pieces of data. It reduces the total overhead lied upon the messages too. In any case, the JMS infrastructure must be tuned accordingly to system needs, considering message persistency, lifetime, etc.

Data unification settles a foundation for effective parallelization in the middle-tier. It notably improves the testing capabilities of the whole system. Presented mechanism partly obliterates the need for distributed caching, an attractive yet highly problematic solution.

Decomposing for Concurrency

Rule engine is the core processing unit of the middle-tier server. A *rule* is a definition of calculation performed on the set of metric values, typically the most up-to-date ones. An example is a threshold rule that verifies if a metric value is within defined boundaries (e.g. temperature) or a mathematical formula where several disparate metric values need to be used to calculate the result. For sake of clarity, we skip all more advanced topics on rule calculations here. The output of the rule can be any action performed by the server, e.g. throwing an alarm, but also an *artificial* metric (created by the server, not monitored devices), which is in turn fed into another rule. Hence, rules form a tree structure which metrics

flow through. The same metric can be obviously used by many rules. As rules are laid out at the startup, it is always known where every original or artificial metric should be passed to.

Rule engine is a classic example of task, data and pipeline decomposition [7]. Each metric is an independent immutable piece of data. Rule is a task, which must be performed every time a set of arguments, metrics, is available. The input metric channel, along with the paths they flow between rules, constitutes a data-driven pipeline processor, a chain of responsibility. The overall structure vows for concurrent architecture.

There are three tiers of processing in that model. The first is an input channel, where immutable metrics are prepared and made available. The second tier is made up of rule *resolvers* (*identifiers*). They place each metric among the arguments of every rule they belong to. Once the argument list is complete, a self-contained *rule job* representing a single rule calculation is created. The last tier is comprised of rule *processors,* which take such jobs and perform associated calculations, likely producing artificial metrics (Fig. 1). As metrics and rule jobs are immutable and independent, numerous rule identifiers and rule processors can act autonomously. It gives the ground for parallelism.

The crucial is however what separates those three tiers. Java 6 Queue implementations (FIFO buffers) constitute the buffering layer [8]. The first layer holds metrics, the second keeps rule jobs, however they are both identically set up. Buffering layer implements the asynchronous Reader/Writer pattern. What has a decisive impact on the performance of the rule engine is the number of buffers, their sizes and the strategy for choosing one to read or write. Long tests proved the following to perform optimally in our setup. The layer is composed of a few buffers (2-4), each has a capacity of about half a million elements. Writers and readers are at least twice as numerous as the buffers. The strategy on reading the data is straightforward: pick the buffer holding the least elements. For writing, use the smallest buffer at the time for a period, e.g. 1000 writings, then repeat choosing. Both round-robin or random choosing strategies appeared to have fallen short of the selected policy. Its efficiency depends on many aspects, some being low level hardware properties (e.g., CPU cache hits).

Having a buffering layer configurable, including various R/W policies, is a precious feature. It greatly helps to find an optimal setup for a given architecture. During the tests it is essential to monitor the contents of the buffers and watch against any missed writings.

Because of the hierarchical composition of rules there is a natural tendency to break calculations into periodic stages, *rounds*. It partly roots in the fear that generating artificial metrics and passing them further may spark off an avalanche of computations. On the contrary, we have found no legitimate reason for staging the processing. The rule engine is left to run at its natural speed continuously. Artificial metrics are not directly handed to the following rules, but fed back into the input channel. It gives a perfectly consistent view on both processing and data. It also results in the highest throughput of all, reaching *100000* non-trivial calculations a second sustained for over a week without a single metric lost.

It is worth mentioning the thread scheduling discipline is typically OS-specific. Therefore a designer must guarantee that multicore CPU will be properly utilized for a given setup. It might need additional tweaking of the OS parameters.

Garbage Collection

Once the architecture is optimized at the level of data structures, algorithms and task parallelization, tuning the garbage collection remains as the pivotal job. Understanding how garbage collectors affect an application is crucial to laying out realistic boundaries on how performing the system can be.

The combination of extensive computing (non-trivial method calls) and frequent memory allocations contributes to numerous garbage collections within a time frame. Moreover, in such case garbage collectors require to operate on those heap structures (e.g. old generation) that inflict stop-the-world pauses - all the application threads get inhibited for a considerable period [9]. Unlike the Java Real Time System, none of four main garbage collectors available in Standard Edition guarantees non-interruptible work for the underlying application. They however differ in the lifecycle of such interruptions and the use of threads to perform on.

Although our system in general exhibits only soft real time requirements, in order to achieve massive continuous processing of metrics it is essential to impose almost hard real time requirements on the JVM. Testing various configurations of GC options for heightened number of calculations, we experienced a common pattern - failing scenario. Once the GC starts pausing the application on a regular basis, it will inevitably break down having less and less chance to make up for the lost time.

Having thoroughly tested our prototype with JRockit, Parallel-Compacting, Concurrent Mark-Sweep and GarbageFirst (G1) collectors in a variety of options, we found the last two establish the most sustainable environment. Due to the nature of processing and uniqueness of our setup such a statement cannot be generalized. It remains only as an incentive for every designer to never neglect the topic.

Table 1: Example Options Used with Tested GC's

GC	Selected options
PC	UseParallelOldGC, DisableExplicitGC, ParallelGCThread, CompileThreshold, MaxGCPauseMillis, NewSize
CMS	CMSIncrementalMode, CMSIncrementalPacing, CMSIncrementalDutyCycleMin, CMSIncrementalDutyCycle
G1	GCPauseIntervalMillis, G1ParallelRSetUpdatingEnabled, G1ParallelRSetScanningEnabled

GarbageFirst is a server-style collector, targeted for multi-processors with large memories. It also well utilizes the 64 bit architecture. G1 is considered the most efficient mechanism available in Java Standard Edition and has been decided a default solution shipped in JDK7.

Garbage collectors undergo constant development, changing their shape across JVM versions. Modern research concepts, as *split bytecode verification* or *lock coarsening,* enhance the overall process.

Memory Allocation

Organising memory allocation is a subject complementary to garbage collection. Java developer is offered to set up at least the boundaries of memory available to the application through a series of JVM options. Besides, the choice of the computer architecture and operating systems notably influences the outcome. As 64 bit multicore servers have become a prevailing standard, excessive use of the heap space emerged tempting. When the 4GB limit per JVM is surpassed (effectively much less depending on the operating system, see Table 2) an application can handle massive volumes of data in memory. The number of threads allowed to be created is also substantially larger. Nonetheless, experimenting with the settings shows it immediately brings a performance penalty in form of extended garbage collections, a hindrance we struggle to overcome at any cost. The other known effect of upgrading to 64 bit architecture comes with the need to operate on extended memory addresses. It is generally considered to moderately hamper the speed of processing.

Table 2: Memory Available in 32 bit Java SE [10]

OS	Heap Size
Linux	2.2-3 GB
Solaris	3.5-4 GB
Windows	1.4-1.8 GB
MacOS	<4GB

As for the garbage collection, choosing the right computer architecture and finally tuning the memory available affects the overall quality of our application. All those topics should be consciously scrutinized.

Profiling

Testing a complex Java system cannot be effectively carried out without the help of profiling tools. A rudimentary support for code orchestration comes with a JDK tool, VisualVM [11]. It allows to watch a lifecycle of the application with respect to its CPU, memory and thread consumption. Monitoring these characteristics is indispensable when tuning the garbage collection and memory allocation properties.

VisualVM and JConsole rely on the JMX support that JVM instance offers. We can easily profit from the feature by exposing some parts of our system as JMX calls. It lets manage the system during tests using a standard out of the box tool.

CONCLUSIONS

Java software design should always be considered with respect to the characteristics and shortcomings of the modern Java runtime environment. Conversely, garbage collection and memory allocation settings are often perceived superfluous, concise choice of data structures and concurrent routines considered secondary. Performance and scalability tests tend to be underestimated.

CERN monitoring and alarm system is a high throughput, semi real-time and most importantly reliable application based on the Java components. Architecting its prototype has brought tangible evidence how some often neglected issues determine the overall result of software development.

We have shortly discussed a selection of topics whose deep understanding may be vital to achieving a quality distributed software. They are more than likely to influence an outcome of designing any Java-based system that shares similar requirements.

REFERENCES

[1] M. Buttner et al., “Diagnostic and Monitoring CERN Accelerator Controls Infrastructure: Diamon Project First Deployment in Operation”, Proceedings of ICALEPCS’2009, Kobe, Japan.
[2] Technical Infrastructure Monitoring, timweb.cern.ch
[3] S. Deghaye et al., “CERN Proton Synchrotron Complex High-Level Controls Renovation”, Proceedings of ICALEPCS’2009, Kobe, Japan.
[4] W. Buczak and M. Misiowiec, “Diamon/Laser prototypes – report”, CERN EDMS, 2011
[5] M. Sobczak, “Programming Distributed Systems with YAMI4”, Lulu, 2010
[6] V. Baggiolini et al., “JAPC - Java API for Parameter Control”, Proceedings of ICALEPCS’2005, Geneva, Switzerland
[7] T. Mattson and B. Sanders and B. Massingill, “Patterns for Parallel Programming”, Addison-Wesley Professional, 2005
[8] B. Goetz and T. Peierls and J. Bloch and J. Bowbeer and D. Holmes and D. Lea, “Java Concurrency in Practice”, Addison-Wesley Professional, 2006
[9] E. Bruno and G. Bollella, “Real-time Java programming with Java RTS”, Sun Microsystems, 2009
[10] Java forum topics, http://stackoverflow.com
[11] VisualVM, http://visualvm.java.net

THIRTY METER TELESCOPE OBSERVATORY SOFTWARE ARCHITECTURE

K. Gillies#, C. Boyer, TMT Observatory Corporation, Pasadena, CA 91105, USA

Abstract

The Thirty Meter Telescope (TMT) will be a ground-based, 30-m optical-IR telescope with a highly segmented primary mirror located on the summit of Mauna Kea in Hawaii. The TMT Observatory Software (OSW) system will deliver the software applications and infrastructure necessary to integrate all TMT software into a single system and implement a minimal end-to-end science operations system. At the telescope, OSW is focused on the task of integrating and efficiently controlling and coordinating the telescope, adaptive optics, science instruments, and their subsystems during observation execution. From the software architecture viewpoint, the software system is viewed as a set of software components distributed across many machines that are integrated using a shared software base and a set of services that provide communications and other needed functionality. This paper describes the current state of the TMT Observatory Software focusing on its unique requirements, architecture, and the use of middleware technologies and solutions that enable the OSW design.

TMT INTRODUCTION

The Thirty Meter Telescope (TMT) is an advanced, wide field (20 arcmin), altitude-azimuth telescope with a primary mirror consisting of 492, 1.44 meter segments. At first light, a facility multi-conjugate adaptive optics (MCAO) system will be available using a laser guide star (LGS) system. The facility's twin Nasmyth platforms will concurrently support multiple instruments, which are all available during the night for observations. Two science instruments will be delivered for use with the MCAO-LGS system: IRIS, a near-infrared instrument with parallel imaging and integral-field-spectroscopy support; and IRMS, an imaging, multi-slit near-infrared instrument. A seeing-limited, wide-field, multi-object optical imaging spectrograph (MOBIE) will also be available at first light. The operations model includes PI-directed observing from remote facilities and on-site service observing by staff. The telescope and facilities are currently in the advanced design phase. The telescope is planned for the summit of Mauna Kea on the island of Hawai'i in the United States.

TMT SOFTWARE ARCHITECTURE

Significant aspects of software systems for large observatories have converged around a few common ideas and solutions due to similarities in the facilities, their requirements for operation and use, and the issues related to development and maintenance [1]. The TMT software architecture takes advantage of these prior solutions when possible. It's then possible to focus attention on the problems unique to TMT and reuse common solutions for the parts of the software system that are known or of little risk. We can also improve upon the solutions used in the previous generation of systems when experience has shown that aspects of the known solutions have issues.

The complexity of some aspects of the TMT software control system scale with the telescope aperture size, and the software must also scale to handle this complexity. Segmented mirror control for TMT requires more moving parts behind the mirror and with it more sophisticated control software than existing segmented mirror systems.

It's also true that not every aspect of TMT software complexity scales with the size of the aperture. Many things that work for 8m class telescopes work just as well with TMT. An example is proposal submission and planning software.

A complete description of the architecture for a system the size of the TMT software system requires more than a few pages. Table 1 shows some of the broad range of challenges related to the software that executes at the telescope site. This paper will show one way the architecture addresses the challenges of Table 1.

Table 1: Architecture Challenges

Challenge	Description
Acquisition	There are demanding requirements on the system performance and coordination for target acquisition and observation setup.
Composition challenges	Some components must be used in a variety of situations or composed in different ways to support different applications. For instance, the MOBIE wavefront sensors must act as part of MOBIE but also work as part of the phasing system.
Wavefront measurement	Wavefront measurement functionality and hardware is potentially spread throughout telescope systems and instruments making it a challenge to cleanly decompose the system and software. This was a problem with 8m class telescopes as well.
Distributed development	Multiple, distributed, international partners provides a new level of software construction and management complexity.
Operations maintenance	The long lifetime of TMT requires an architecture that can be enhanced and modified without impacting the operations system.

#kgillies@tmt.org

Architecture Overview

TMT has adopted the idea of a technical architecture and functional architecture from other recent large telescope projects (e.g. ALMA [2], ATST[4]). The *technical architecture* is the software infrastructure that provides the foundation for the functional architecture. System features, such as logging and command support, are part of the technical architecture. The *functional architecture* consists of the decisions and software components that enable the activities of the observatory from the point of view of the users. For instance, how does the system collect header information for a science data frame?

A software system should be viewed in many ways. At the highest level in the functional architecture, TMT software is modeled as the 5 large *principal systems* shown in Figure 1. Each principal system is focused on specific, easily-identifiable functionality, and each is itself a collection of other software components or systems. Viewing the system at this level allows one to think more easily about flow of control and where software functionality exists within the system.

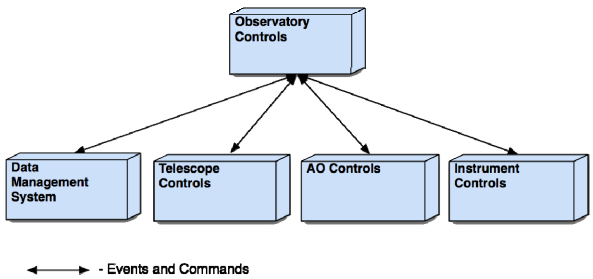


Figure 1: One view of the TMT software architecture splits the system into five large principal systems.

This view shows communication is hierarchical and flows down from Observatory Controls to the other principal systems. This command communication is low-bandwidth by design; any high-speed communication occurs within a single principal system.

Figure 2 drills down one more level to show sub-systems that represent major hardware and software components within the principal systems. At the top of the figure are the observing user interfaces and the software components within Observatory Controls that sequence and synchronize the activities of the other systems to generate the user's desired science data.

The integration of these software components requires software infrastructure that is outside the scope of the individual components themselves. The horizontal bar dividing Figure 2 in half represents this software infrastructure. The idea of shared software infrastructure based on a set of services and associated software focused on integration is a successful strategy in large observatory software architecture [1, 2, 4]. TMT calls this software Common Software (CSW). CSW is the implementation of the technical architecture.

Common Software is a collection of software and services. A *service* is a set of related software functionality together with behavior and the policies that control its usage. TMT CSW uses external packages (i.e., software not developed by TMT–COTS or open source middleware, etc.) to implement the CSW services. Abstractions and wrappers are present between the CSW services and the external packages to enable some isolation from specific product choices. For a component programmer integration of a component with TMT requires the use of a service-oriented API and library code that must be linked with the component.

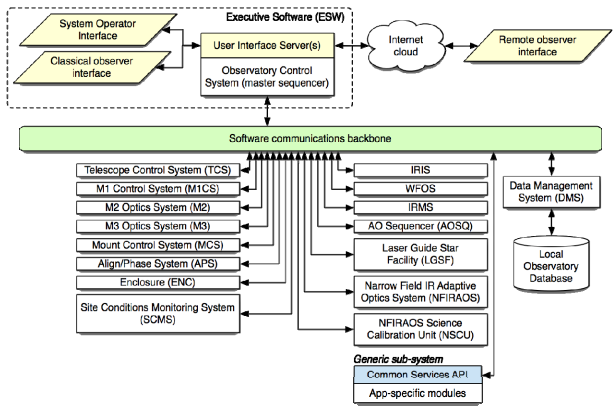


Figure 2: The major systems of the Telescope Controls, AO Controls and Instruments connected by the software infrastructure provided by the technical architecture.

Table 2 is a list of services provided by CSW that are needed to enable the functionality of the Functional Architecture.

Table 2: List of Planned CSW Services

Service	Task Description
User single sign on	Centrally manage user authentication and access control
Commands	Support for subscribing to, receiving, sending, and completing commands in the form of configurations
Location/Connection	Locate and connect to components within the distributed system
Events/Telemetry	Enable event-based functionality based on publish, subscribe paradigm
Alarm/Health	Support monitoring and publishing component alarm and health signals
Configuration	Manage initial values and system and component configurations history
Logging	Capture and store system logging information
Time	Standards-based, precision time access

One important example is the event services. These services allow one component to send a piece of information (i.e., an event) to one or more other

components. The Event Service provides a high-performance, publish-subscribe message system. Components can signal actions (events), publish status information (telemetry), or publish control information (event streams). One big advantage of this type of service is that the publishers and subscribers are decoupled; each requires no knowledge of the existence of the other.

OBSERVING MODE ORIENTED ARCHITECTURE

Operations experience has shown there are drawbacks to limiting the modeling and construction of the software structure to the principal system level, and this has driven architecture changes for TMT. For instance, principal systems are large systems that must handle all observing modes and operations scenarios. This results in broad, complex software interfaces that are difficult to verify and modify during operations. It continues to be valuable to view the system at the principal system level, but to address this and the challenges of Table 1 a more flexible approach is required. Flexibility is a key design concept that can result in a software system that can respond to the changing needs of science operations over the project lifetime.

The functional structuring approach planned for TMT that addresses these issues is called Observing Mode Oriented Architecture (OMOA) [3]. An observing mode is a well-defined instrument observing task and an associated set of owned resources, procedures, and capabilities that implement the mode. An example observing mode for TMT is: IRIS multi-filter integral field spectroscopy using the NFIRAOS adaptive optics unit with AO laser guide star correction. An instrument will generally have several associated observing modes for acquisition, science objects, and calibrations.

A goal of this architecture is to eliminate software waste and run as little software as is necessary to execute an observation using a specific observing mode. To accomplish this, the software within principal systems must consist of smaller components that are then assembled into a dynamic system configuration that is specific to the observing mode.

To explain this approach, the software components you might find in a principal system are shown as layers in Figure 3 with specific responsibilities described in the following sections.

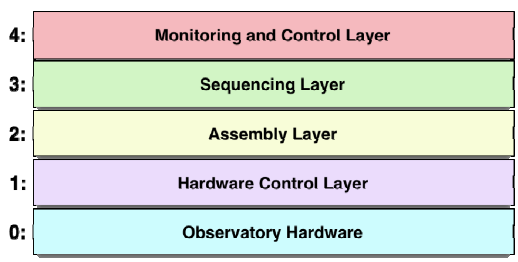


Figure 3: OMOA software structure layers.

Hardware Control Layer

In 2011 the trend is towards motor controllers and other hardware controllers and sensors that are network-resident devices capable of controlling multiple channels or to Programmable Application Controllers communicating via high-level commands over a standard TCP/IP-based network. It is generally no longer necessary to develop single use, low-level device drivers; a software investment that requires skilled programmers and significant effort that generates long-term technical debt and hinders change during operations.

The lowest layer in the OMOA software system, called the Hardware Control Layer, consists of all the controllable hardware that is available for use by higher levels of software. A sea of similar software components called Hardware Control Daemons (HCD) at layer 1 controls the TMT low-level hardware of the telescope, adaptive optics, and instruments.

Each HCD is associated with one or more networked motion controllers or other low-level hardware controllers (shown as layer 0 in Figure 3). The HCDs act as adapters and provide a uniform software interface and feature set focused on device control to the layers above. The HCDs are *always executing,* and each can be accessed at any time by the software layers above.

This layer is one place where external systems can be integrated. As an adapter, a HCD can isolate a proprietary connection to an external system making it look like a conforming system device. The HCD also provides a suitable location for device simulation allowing device end-to-end system testing without hardware presence.

Assembly Layer

The Assembly Layer exists just above the Hardware Layer at layer 2 in Figure 3. Software at this layer consists of components called *Assemblies* with two roles in the OMOA. The first role is to allow the grouping of HCDs into higher-level entities. This is required when individual hardware devices must be considered as a unit or requiring processing. The second role of components in the Assembly Layer is to provide more sophisticated hardware control functionality that integrates devices across different HCDs to produce higher-level devices or add uniformly useful capabilities.

Assemblies can be transient or long-lived. An example of a long-lived Assembly is one that provides telescope pointing, tracking, and offsetting. An Assembly can also be created dynamically to provide combinations of HCDs that need to be coupled for a specific observing mode during an observation. An example might be the coordination of wavefront sensor detector readout processing and the control of the probes for the wavefront sensors.

Sequencing Layer

The Sequencing Layer is layer 3 in Figure 3. Components at this level are called sequencers because they control and synchronize the actions of the HCDs and Assemblies. The sequencer components are dynamically

created and composed in order to execute a specific observing mode. This approach is possibly the most innovative part of this software architecture, because it is this layer that gives the software its compositional flexibility and other qualities. Individual sequencers can provide higher-level control of a set of distributed hardware. The ability to cut across the development hardware boundaries and compose and control hardware as needed is what allows this approach to eliminate many of the challenges mentioned in Table 2.

The components in this layer share software interfaces that allow them to be plugged together to form the sequencing engine for a specific observing mode. There can be one or many sequencing components in an observing mode sequencer. The goal is that the sequencing components for a mode be assembled into a single process. This minimizes complexity and performance issues related to distributed processes and communication of commands across process and machine boundaries. A large amount of software related to command input/output in systems is eliminated, and fewer tiers result in higher performance and simplicity.

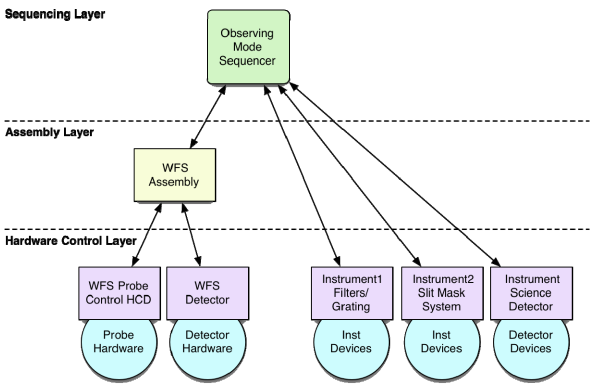


Figure 4: A simple example showing the lowest 4 layers of an OMOA observing mode configuration.

The sequencer for an observing mode is constructed during observation execution with a *Configuration Factory* that takes as input an observing mode and an observation description from another software tool such as a high-level planning GUI. The Configuration Factory has been programmed with instructions on how to construct the matching sequencing process for the observing mode.

Figure 4 shows a simplified, partial example of an instrument like TMT's MOBIE in a seeing-limited observing mode. At the lowest level are the HCDs for the wavefront sensor probes, instrument hardware, and detectors. An assembly exists to integrate and provide higher level functions for the wavefront sensor hardware and detector. All other sequencing for the observing mode is in the Observing Mode Sequencer that is created dynamically for the observation. The HCD components could be separate processes running on distributed hardware while the sequencing components exist in a single process within the Observatory Controls.

This example demonstrates the flexibility inherent in this approach. It allows the grouping of software and hardware in an optimal arrangement for a specific observing mode with only the functionality needed to support a specific mode. During operations this allows new observing modes to be rolled out more easily with minimal influence on current functionality.

Monitoring/Control Layer

The Monitoring/Control Layer (layer 4 in Figure 3) is the layer of software that contains the user interface programs that are used to observe with the telescope. At TMT there will be graphical user interfaces for use by observers during the night. These applications use the CSW services to control and monitor the system.

SUMMARY

The TMT software architecture is similar to the systems constructed for 8m telescopes, but it has been enhanced to take advantage of changes in software and hardware technology as well as 10 years of experience with the principal system architecture approach.

The TMT technical architecture is based on a set of shared software services, each of which is itself based on open-source or commercial software.

The functional architecture uses the structuring approach of the Observing Mode Oriented Architecture. The goal of OMOA during observation execution is to allow, for any given observing mode, the minimal amount of software needed to execute the mode. By taking this approach, many of the problems outlined in Table 1 are minimized or eliminated.

The innovation in OMOA is the implementation of the principal systems as more focused fine-grained components at the architectural level. This single change provides the opportunity to reduce the amount of software needed at the telescope, thereby reducing the complexity of the runtime system. This, coupled with the use of dynamic system configurations focused on executing individual observing modes, addresses the need to sequence systems with more flexibility and higher performance than currently possible.

REFERENCES

[1] K. Gillies, J. Dunn, D. Silva, "Defining common software for the Thirty Meter Telescope," Proceedings of SPIE Vol. 6274, 62740E (2006).

[2] J. Schwarz, G.Chiozzi, P. Grosbol, H. Sommer, D. Muders, ALMA Software Architecture, http://www.alma.nrao.edu/development/computing/docs/joint/draft/ALMASoftwareArchitecture.pdf.

[3] K. Gillies, S. Walker, "An observation execution system for next-generation large telescopes," Proceedings of SPIE Vol. 7740, 7740Q (2010).

[4] S. Wampler, B. Goodrich, "ATST Software Concepts Definition," Document SPEC-0013, Rev B, http://atst.nso.edu/files/docs/SPEC-0013.pdf

TOWARDS A STATE BASED CONTROL ARCHITECTURE FOR LARGE TELESCOPES: LAYING A FOUNDATION AT THE VLT

R. Karban, N. Kornweibel, ESO, Garching, Germany
D. Dvorak, M. Ingham, D. Wagner, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA.

Abstract

Large telescopes are characterized by a high level of distribution of control-related tasks and will feature diverse data flow patterns and large ranges of sampling frequencies; there will often be no single, fixed server-client relationship between the control tasks. The architecture is also challenged by the task of integrating heterogeneous subsystems which will be delivered by multiple different contractors. Due to the high number of distributed components, the control system needs to effectively detect errors and faults, impede their propagation, and accurately mitigate them in the shortest time possible, enabling the service to be restored. The presented Data-Driven Architecture is based on a decentralized approach with an end-to-end integration of disparate, independently-developed software components. These components employ a high-performance standards-based communication middle-ware infrastructure, based on the Data Distribution Service. A set of rules and principles, based on JPL's State Analysis method and architecture, are use to constrain component-to-component interactions, where the Control System and System Under Control are clearly separated. State Analysis provides a model-based process for capturing system and software requirements and design, greatly reducing the gap between the requirements on software specified by systems engineers and the implementation by software engineers. The method and architecture has been field tested at the Very Large Telescope, where it has been integrated into an operational system.

CONTEXT

The European Southern Observatory (ESO) carries out an ambitious programme focused on the design, construction and operation of powerful ground-based telescopes for astronomy. Worldwide, Extremely Large Telescopes are considered one of the highest priorities in ground-based astronomy.

The European Extremely Large Telescope (E-ELT), a revolutionary new 40-metre-class ground-based telescope, will be the world's biggest eye on the sky, when it begins operating early in the next decade [1].

The E-ELT is characterized by a high level of distribution of control-related tasks and will feature diverse sets of data flow patterns and large ranges of sampling frequencies.

Within the E-ELT, the Telescope Control System (TCS) will maintain wave front quality throughout the duration of the observation. Considering the multitude and inter-relation of E-ELT distributed control loops (covering about 1000 moveable mirrors, and actuator stroke management), maintaining the wave front during the observation is foreseen to be more demanding than acquiring the target. This is a fundamental difference from our past experience on the Very Large Telescope (VLT) project - the essential complexity (the complexity associated with the underlying problem) has increased, and thus it is imperative to avoid adding any more complexity than necessary.

The TCS includes all hardware, software, and communication infrastructure required to control the telescope (including the dome), and to interface to subsystems down to, but not including, actuators and sensors, as they are typically delivered together with the respective sub-system. The main functions provided by the TCS are related to Control and Interlock (an automatic and immediate stop of a unit in case of a safety critical situation) handling, and fault management.

The **main challenges** can be summarized as follows:

- Large number of control points (the primary mirror alone encompasses 15000 actuators)
- Number of interfaces (15 subsystems, 9 focal stations, site operation)
- Large data volume (700 Gflops/s, 17 Gbyte/s in real-time in adaptive optics) of engineering data
- Multitude of interacting, distributed control loops (from 0.01Hz up to kHz rates)
- Software intensive distributed control strategy
- Integration of heterogeneous distributed components, provided mainly by contractors, adds additional complexity.

ARCHITECTURE

Separating the concerns of astronomy from those of technical implementation is a key goal of the architecture. Astronomy domain-specific knowledge should be isolated from the (technical) subsystem domain, which is typically contracted to an expert in that domain. The implementers of the subsystem should not be required to have knowledge of the astronomy domain. Another major architectural driver is the need to query knowledge of the state (e.g., temperatures, position of actuators, and taking into account uncertainties of sensor readouts) of the physical system and to control this state. Due to its high distribution and tight coupling this knowledge is essential to be able to deliver a functioning system.

The architecture must satisfy certain (domain independent) quality goals. The most important are

related to the conceptual integrity. The conceptual integrity is the underlying vision that unifies the design of the system, at all levels. The architecture should do similar things in similar ways.

The **main goals** for the TCS **architecture** are:

- Define a framework and design rules to enable the software engineer to develop the domain specific applications which address and match the functional needs from the wave front control strategy.
- Contain system complexity.
- Promote modifiability and scalability (and long-term maintainability).
- Enable high availability and fault tolerance.

These strategic goals can be achieved by applying certain tactics:

- Carry out the system design according to a well defined set of design patterns.
- Have a uniform way of designing closed loop control on the subsystem actuators, i.e. how are the sensor data processed, the state of the hardware system estimated, and the actuators commanded?
- Represent the state of the hardware system and the control system in a uniform way.

The design of the control system architecture brings together engineers from different domains: system, software, control, and electronics. Close collaboration between those disciplines is essential to close the gap between systems engineering and the other technical disciplines.

The data-driven architecture (explicitly formalizing the data and meta-data produced and consumed by a system, transported with messaging [10], see Figure 1) consists of three tiers organized in a decentralized manner in order to separate domain knowledge (e.g., tracking a celestial object vs. mirror control), integrate heterogeneous control systems, and support the implementation of flexible wave front control strategies.

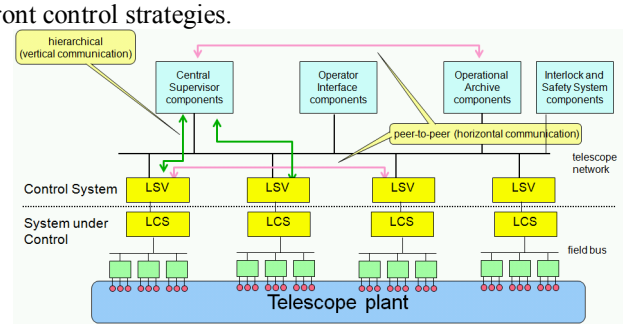


Figure 1: Conceptual Architecture.

The first tier provides robust and simple access to sensors and actuators of a subsystem and is called the Local Control System (LCS). The second tier adapts the LCS to the astronomy domain and is responsible for closed loop control of the sub-system's actuators and sensors. It is called the Local Supervisor (LSV). The third tier, the Central Supervisor, provides high level interaction with users and among subsystems.

STATE ANALYSIS AND MBSE

A Model Based Systems Engineering (MBSE) methodology is characterized as the collection of related processes, methods, and tools used to support the discipline of systems engineering in a "model-based" or "model-driven" context [7]. The State Analysis (SA) method [3] [8] is targeted to the control related domain, and focuses on behaviour, which is an often underestimated aspect in Systems Engineering. The SA method, which is founded on a state-based architecture and goal-based operation, defines a process for identifying and modelling the states of the physical system and their relationships. The methodology enables effective coordination, robust execution, and flexible response mechanisms in the system by defining goals (e.g., keep tracking error within a certain RMS error) for the states of the physical system instead of defining precise execution procedures up front.

State Analysis provides a uniform, methodical, and rigorous approach for developing control system architecture by:

- Discovering, characterizing, representing, and documenting the state variables of a system;
- Modelling the behaviour of state variables and relationships among them, including information about hardware;
- Identifying interfaces and operation; and
- Capturing the mission objectives in detailed scenarios motivated by operator intent.

In State Analysis, nominal and off-nominal states are given equal stature in the model. State Analysis provides the opportunity for (but does not obligate) earlier consideration of faults and off-nominal behaviour. Fault Detection, Isolation, and Recovery (FDIR, also known as Fault Management) becomes an integral part of the control system design. A strict boundary is drawn between the **Control System** being designed and the target **System Under Control**, as shown in Figure 2.

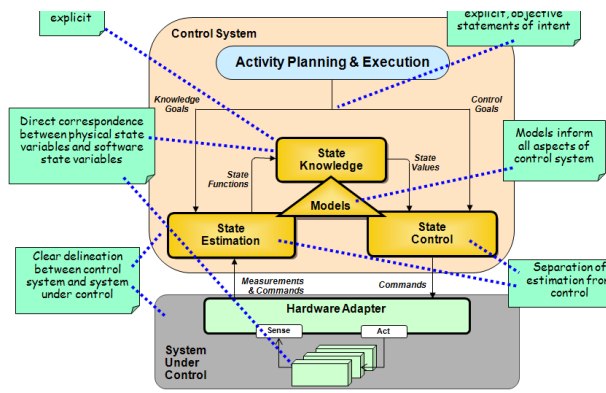


Figure 2: State Analysis Architectural Concepts.

State Analysis provides sound architectural principles and rules which help to achieve the requirements and goals of the E-ELT TCS. Thus, many concepts of the method have been baselined by the project, and adapted to the specific needs of the project (like where to draw the line between

control system and system under control for contractual reasons).

VLT FIELD TESTING

As part of validating the technology decisions for the E-ELT, control systems at the VLT (Figure 3) [4] are being refurbished using technologies intended for use in the E-ELT control systems. The upgrades serve several purposes: field test technologies and methods in an operational environment (outside the lab), provide input into the E-ELT technology decisions, address obsolescence in the UT telescope control systems, and ready observatory technical staff for the construction of the E-ELT.

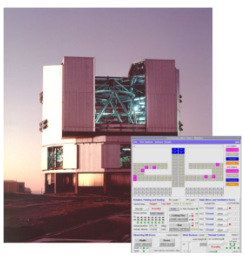


Figure 3: VLT UT.

As the first step in the upgrade and field test program, the enclosure (dome) control system (ECS) of one Unit Telescope (UT) was refurbished in 2010, using the SA method and architecture. It is now successfully operating at the Paranal observatory.

The Enclosure Control System Upgrade comprised:

- Dome and Windscreen;
- Seal, louvers, Observing Doors;
- Telemetry
- Thermal input/output (I/O);
- Approximately 1500 I/O points.

The refurbished control system interfaces to existing sensors and actuators in the field, and integrates transparently with the existing dome control interface at the supervisory level. The application of SA was therefore limited to the LCS and LSV.

The SA control pattern drove the development of the data model, which consisted of building a façade to the System Under Control, identifying and defining State Variables (SVs) and specifying Goals).

Though only a limited and introductory application of SA was made, the architectural rules of SA aided in improving the design of the control system software. Estimators (which generate state knowledge based on available evidence) read blocks of field data and publish data to State Variables. The Estimators were largely implemented with LabView.

During 2011 and 2012, the project will be working to upgrade the control system of the main axes. The goal of the project is to achieve the same or improved scientific performance while tackling obsolescence in the system by upgrading with specific technologies and architectures targeted for use in the E-ELT.

The main axes control system provides the means for telescope positioning (moving the telescope in either Altitude or Azimuth axis in order to point the telescope) and telescope tracking (coordinating the movements in both axes in order to track celestial objects given their position, motion and the current time).

The main axes upgrade applies more rigorously the SA method than the enclosure upgrade, together with other MBSE concepts. In particular, a SysML [5] profile for SA is being developed, allowing integrating the SA artifacts into an overall system model. The overall system model is built according to principles of the Object Oriented Systems Engineering Method (OOSEM [5]), which distinguishes a logical and physical model of the system, and also provides means to describe the system "as-is" and "to-be" and the relations between the two– particularly useful for an upgrade project. JPL has already defined an initial UML profile for SA [6], which serves as a starting point for the SysML profile.

Figure 4 shows a snippet of a State Effects Diagram (SED), which is a SA artifact used to identify SVs, Measurements, Commands, and the causal effects between them. The main focus in this diagram is the set of effects on and of the physical state variable "Azimuth Position And Velocity". Every box on the diagram corresponds to a physical SV which has to be estimated and possibly controlled.

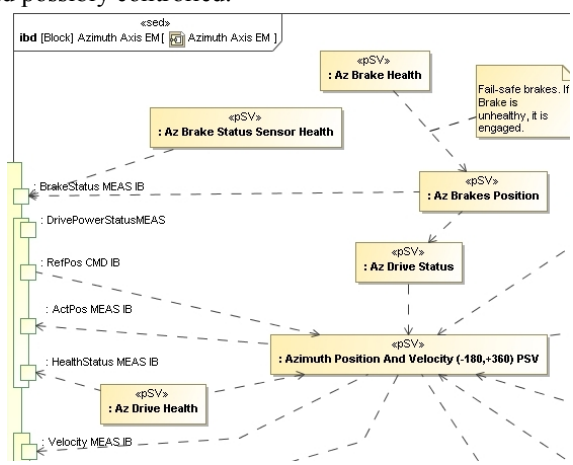


Figure 4: State Effects Diagram of Main Axes.

The abstract <<affects>> relationships are later on modelled in a more rigorous way using SysML State Charts, Parametric models or Activities, if and only if a more accurate description is needed.

The corresponding conceptual architecture (Figure 5) can be derived from the SA artifacts, including the SEDs. It shows two SVs "EncHeadHealth SWSV IB" and "AzPosVel SWSV IB" with their respective estimators and controller, interacting with the Azimuth Axis Local Control System. Since everything is contained in the same model, referring to the same model elements, the information on the system is all consistent.

Many artifacts created using SA can be re-used across projects and systems because they express domain concepts in an implementation-independent way. For example, SA artifacts from a previous design of an Antenna array [2] can be leveraged for the E-ELT.

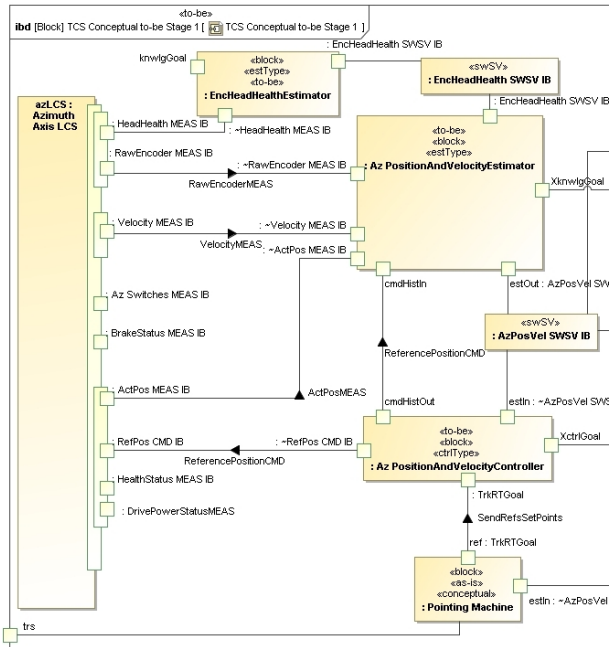


Figure 5: Conceptual State Based Architecture.

IMPLEMENTATION

JPL implemented the State Analysis method with its Mission Data System (MDS) architecture [9], which provides software libraries, classes, and behaviour reflecting the SA concepts (e.g., state variables, goals, etc). The current MDS implementation is based on method invocation: for example, a state variable is queried if its current constraint (its goal) can still be satisfied.

The current ESO prototype implementation is based on messaging, using the Data Distribution System (DDS) as its middleware. State Variables are realized by DDS topics. Consequently, controllers and estimators can simply publish and subscribe to a SV, making combination of multiple SVs in a control algorithm simple and without requiring knowledge of where the SV resides or what estimates it.

In this way, the full SA control pattern is implemented with the publish/subscribe mechanism, which allows runtime attaching and detaching of clients, for example to trace measurements and verify the behaviour of the control system.

SUMMARY AND FUTURE WORK

State Analysis is built on a sound theory which enables building an architecture for a distributed system, like the E-ELT, following well defined principles and rules. Although the scalability for very large systems needs still to be verified, the first practical experiences on the VLT give us confidence to pursue the path of applying SA for the VLT upgrade and eventually for the E-ELT control system.

Integration with other MBSE practices is fundamental to address all relevant system aspects, and requires, in particular, a SysML profile. The definition of the mapping between SA and SysML, and the formalization of SA concepts and relations in an ontology is part of the ongoing collaboration between ESO and JPL.

This research was carried out at the European Southern Observatory and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] R. Gilmozzi, J. Spyromilio, „The 42m European ELT: status"", Proc. SPIE 7012**,** 701219 (2008)

[2] J. Choi, A. Coleman, D. Dvorak, J. Hutcherson, M. Ingham, C.Y. Lee, P. Wolgast, "Goal-Based Operations of an Antenna Array for Deep Space Communication", iSAIRAS. Los Angeles, CA. Feb 2008

[3] M. Ingham, R. Rasmussen, M. Bennett, A. Moncada, "Engineering Complex Embedded Systems with State Analysis and the Mission Data System", AIAA Journal of Aerospace Computing, Information and Communication. Vol. 2, No. 12, December 2005, pp-507-536

[4] K. Wirenstrand, "VLT telescope control software: status, development, and lessons learned", Proc. SPIE 2003, vol. 4837, p. 965 (2003(

[5] S. Friedenthal, A. Moore, R. Steiner, "A Practical Guide to SysML", Morgan Kaufmann OMG Press, 2009

[6] A. Murray, R. Rasmussen, "A UML Profile for State Analysis," IEEE Aerospace Conference, March 2011

[7] J. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Rev. B, INCOSE Technical Publication, INCOSE-TD-2007-003-012008

[8] G. Jones, D. Wagner, D. Dvorak, A. Mishkin, "Human-Rated Automation and Robotics", JPL Technical Report D-66871, 2010

[9] Mission Data System, http://mds.jpl.nasa.gov

[10] G. Hohpe , B.Wool, "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions", Addison-Wesley Longman, 2003

MIDDLEWARE TRENDS AND MARKET LEADERS 2011

A. Dworak, P. Charrue, F. Ehm, W. Sliwinski, M. Sobczak, CERN, Geneva, Switzerland

Abstract

The Controls Middleware (CMW) project was launched over ten years ago. Its main goal was to unify middleware solutions used to operate CERN accelerators. An important part of the project, the equipment access library RDA, was based on CORBA, an unquestionable standard at the time. RDA became an operational and critical part of the infrastructure, yet the demanding run-time environment revealed some shortcomings of the system. Accumulation of fixes and workarounds led to unnecessary complexity. RDA became difficult to maintain and to extend. CORBA proved to be rather a cumbersome product than a panacea. Fortunately, many new transport frameworks appeared since then. They boasted a better design and supported concepts that made them easy to use. Willing to profit from the new libraries, the CMW team updated user requirements and in their terms investigated eventual CORBA substitutes. The process consisted of several phases: a review of middleware solutions belonging to different categories (e.g. data-centric, object-, and message-oriented) and their applicability to a communication model in RDA; evaluation of several market recognized products and promising start-ups; prototyping of typical communication scenarios; testing the libraries against exceptional situations and errors; verifying that mandatory performance constraints were met. Thanks to the investigation performed the team have selected a few libraries that suit their needs better than CORBA. Further prototyping will select the best candidate.

CERN MIDDLEWARE

The Controls Middleware (CMW) project was launched at CERN over ten years ago. Its main goal was to unify middleware solutions used to operate CERN accelerators. Many software components were developed, among them the Remote Device Access (RDA) [1] library. The main responsibility of the library was to allow communication with servers that operate hardware sensors and actuators. The RDA design corresponds to the Accelerator Device Model [1] in which devices, named entities in the control system, can be controlled via properties. RDA implements this model in a distributed environment with devices residing in front-end servers that can run anywhere in the controls network. It provides a location-independent and reliable access to devices from control programs. By invoking the device access methods, clients can read, write, and subscribe to device property values. Currently over 4000 servers (processes) are deployed, which contain altogether almost 80,000 devices. In total the system gives access to more than 2,000,000 properties/IO points, on which clients may perform read/write operations or monitor their values. [2]

Present Implementation

From the beginning there were certain requirements [3] imposed on RDA that drove its implementation: relying only on standards; interoperability with the already existing communication infrastructure at CERN; portable on LynxOS with an old gcc v.2.95 compiler, Linux, Windows, HP-UX and AIX (only the first three are still supported; LynxOS is being eradicated); C/C++ and Java bindings for client/server libraries; request-reply and publish-subscribe operations on device data. Each call type should provide timeout settings and handling of communication errors. Moreover, complementary, centrally managed services like naming service, reservation service and access control should be supplied.

To facilitate development of the new library it was decided to base it on an already existing, mature product. CORBA [4] was a very popular middleware at that time and fulfilled all the requirements. Thus it was chosen as the communication layer. The C++ implementation was based on omniORB (currently 4.1.2,) and the Java implementation on JacORB (currently 2.2.4.) RDA library wrapped CORBA, hiding all its complexities and providing a simple to use API. The proposed solution was widely accepted and became an operational and critical part of the infrastructure.

Shortcomings of the System

Unfortunately, the demanding run-time environment revealed a few shortcomings of RDA. Accumulation of fixes and workarounds led to unnecessary complexity. Desire to deliver a better, more user-friendly solution led to a general review of the system. Discussions with library clients helped to identify several major issues, of which the most troublesome are the ones directly correlated with CORBA [5]. First, the CORBA standard is inherently huge and complex. Libraries that try to fully implement it have a major memory footprint. This is an issue especially for older front-end computers. It is well understood that RDA as a communication framework uses only a small fraction of the CORBA platform, but users still have to pay the full run-time price. On the other hand, libraries such as JacORB do not implement the full functionality. This leads to mismatches in behaviour of Java and C++ bindings. The struggle to support "asynchronous" operations on top of the synchronous calls leads to unnecessary complexity in the library code and design. Second, the way CORBA is used in RDA leads to multiple data conversions between different representations. This is both time consuming and leads to higher memory usage. Third, CORBA is based on the static Interface Definition Language (IDL), which is difficult to manage and evolve in large, complex environments such as CERN. Finally, the community supporting open-source implementations is shrinking.

There is a significant lack of new releases from the major implementations like JacORB, even if the major bugs have been identified and fixed a long time ago.

MIDDLEWARE EVALUATION

In view of a 1-year accelerator shutdown at CERN, starting end of 2012, there is a unique opportunity for introducing a major new version of RDA, which should solve all the limitations experienced with CORBA. Therefore, the CMW team launched the middleware review process, aiming at choosing a new, modern middleware library, to be used for the future version of RDA.

In addition to previously specified general requirements we expect that the new transport library provides:

- Consistent implementation for C++ and Java.
- Easy to trace peer-to-peer communication with reliable request/reply and publish/subscribe messaging patterns.
- Synchronous and asynchronous communication
- Quality of Service (QoS): timeout management, message queues and priorities, various thread management policies.
- Small library size, low memory and resource usage.
- Certain performance characteristics (described later)
- No, or only a few, external dependencies that can be linked with an application, preferably no need for additional services (e.g. brokers, global servers, daemons).
- Open source, with a license allowing to redistribute our product further; good documentation, and support from a large active community.
- Simple, easy to learn and use API.

The CMW team evaluated several market recognized middleware products. A short description of each product is provided below, including a general assessment and results of tests. Detailed performance results and other quantitative measurements are gathered and presented in the next paragraph. All opinions and criticism are based only on our knowledge and products evaluation.

In line with the requirements the following middleware standards and protocols were of no interest: XML-based protocols (e.g. SOAP, XMPP), Stomp, P2P (FastTrack, BitTorrent), MPI, MQTT (rsmb, Mosquitto) nor WebSphere MQ.

The Current Solution: omniORB/JacORB

CORBA is an object-oriented communication platform created by OMG. The standard defines the wire protocol and the IDL, which is used to specify object interfaces. It describes also mappings from IDL to several languages. The complexity of the communication process is hidden from the user, who cannot differentiate between a local and a remote call. The standard and chosen implementations are well documented. Unfortunately there are many shortcomings described in the previous paragraph. Also, the CORBA API is old-fashioned and heavy, thus it has a very steep learning curve and its community shrinks.

Evaluation of Ice

Ice [6] belongs to the object-oriented middleware category. It is conceptually very similar to CORBA, which is an advantage for those who already know it.

The product supports C++ and Java, and runs on Linux and Windows. Compilation on LynxOS fails due to the use of modern C++. It has a static type system and relies on separate specification files to describe interfaces and data structures. Apart from a request-reply model, Ice provides a publish-subscribe event distribution service called IceStorm. Full control over QoS and many tuning options are available. Performance wise Ice satisfies our needs. It uses a compact binary encoding that conserves bandwidth and is very efficient to marshal and unmarshal. Additionally protocol compression can be enabled. Sizes of statically compiled libraries and of binaries of a simple ping-pong server and client indicate a heavy use of global state that brings in the majority of Ice, no matter how much of it is actually used. On the other hand, the well designed API, modern and flexible IDL, easy to use language mappings, up-to-date documentation and a detailed tutorial are a big plus. The library is distributed with GPL license; sources are available for download.

Ice seems to be a very strong candidate due to its industrial presence and number of existing deployments. It also fulfils majority of our requirements.

Evaluation of Thrift

Thrift [7] belongs to the service-oriented middleware category, which means that the central notion in this system is that of remote services being accessed over the network.

The library supports C++/Java and runs on Linux/Windows. Compilation for LynxOS is problematic due to the use of modern C++ features. Thrift has a static type system and relies on separate specification files to describe the service interface and data structures. It supports simple request-reply communication in synchronous and asynchronous mode. It has a small memory footprint and fulfils the performance needs, but it is still an immature product with an incomplete implementation. Tutorial on the product webpage is empty and there is no documentation.

We decided to exclude Thrift from further investigation.

Evaluation of ZeroMQ

ZeroMQ [8] is a message-oriented middleware library, which resembles the standard Berkeley sockets. Because of supported communication patterns and various transports like in-process, inter-process, TCP and multicast it may be easily used as a concurrency framework.

The core of the library is written in C. Bindings for C++, Java (through JNI) and many more languages are supported. The library runs on most modern platforms.

With minor changes it is possible to run it on LynxOS. ZeroMQ has no type specification and does not know anything about the data a user sends. For this reason it has to be used with an external serializer. Because of similarities to the BSD sockets the API is familiar and easy to learn and use. In contrast to the BSD, the ZeroMQ API is more intuitive and user-friendly. Moreover, apart from simple socket send/recv calls, many complex communication patterns are implemented and ready to be used (e.g. request-reply, publish-subscribe, workload distribution). Users have full control over communication policies and QoS (synchronous or asynchronous communication, timeouts, high water marks). The library has a small memory footprint. To achieve the best possible performance it uses different protocols depending on the peers location (TCP, PGM multicast, IPC, inproc shared memory). Parallel protocols may be easily changed so an eventual upgrade from unicast to multicast is easy. The direct connection between the system parts results also in reduced maintenance costs as there is no need for brokers or daemons. A detailed documentation and broad, easy to follow tutorial are available on the product website. The project is under the LGPL license, with a large and active open source community. If needed, full commercial support may be obtained from iMatix, the authors of the product.

We consider ZeroMQ as one of the major candidates to replace CORBA.

Evaluation of YAMI4

YAMI4 [9] belongs to the message-oriented middleware category, in which communicating peers exchange messages between each other. The distribution is therefore explicit and seen in the user code.

The library supports C++/Java and runs on Linux/Windows. With small changes it is possible to compile it for LynxOS. YAMI4 has a dynamic type specification. Data structures (messages) are created dynamically without describing them with IDL. It is an inherently asynchronous communication system with support for request-reply and publish-subscribe over TCP. QoS may be configured through message priorities and timeouts. The library has a small memory footprint and, as our tests show, even if considerably slower than the statically typed products, it fulfils the performance needs. It is an open-source project under GPL, with a thorough documentation and a modern, intuitive API.

YAMI4 is already successfully used at CERN. Unfortunately, community behind the product is small.

Evaluation of the DDS Products

DDS [10] (Data Distribution Service) is an OMG standard, targeting real-time distributed systems. It belongs to the data-oriented middleware category, where the communicating parties declare their interest in a topic and the system takes care of delivery of only relevant data.

There are five wire-interoperable implementations of DDS. We evaluated the three most mature ones. All three products support C++ and Java languages, however due to use of modern C++ they do not support LynxOS out of the box. DDS has a static type system and relies on separate specification files to describe data structures. Compatibility of the generated code with the code generated from the CORBA IDL may be accomplished. Single-direction data flow is the most frequent use-case. It is possible to set up request-reply communication but this requires two symmetric channels. Because of the nature of the channels, this approach is not applicable for CMW, thus additional request-reply middleware would have to be used in parallel. DDS is an asynchronous system that supports many QoS settings, including message priorities. A nice additional feature is Dynamic Discovery, which allows a DDS application an automatic discovery and connection with another DDS application. This feature does not work for us as our network do not support multicast. The products are well documented, but the DDS API is neither easy to use nor compact. In fact, the multitude of settings and concepts provided by the standard is overwhelming and renders the products to be cumbersome and difficult to use.

Evaluation of OpenSplice DDS

OpenSpliceDDS [11] is the only DDS implementation that needs a separate daemon process on each node as individual user processes do not use the network services directly. The daemon is used for service discovery and for data transfer between nodes. Such a solution creates additional complexity, which should be avoided in CMW.

Evaluation of CoreDX DDS

CoreDX [12] is a small-footprint DDS implementation. Unfortunately, due to the licensing policy, further redistribution to third parties would be problematic.

Evaluation of RTI DDS

RTI [13] provides the most mature and widely adopted implementation of DDS. It is distributed with a number of useful tools for system monitoring and administration. As a research organization, CERN is eligible for a free of charge IRAD license and even access to the source code is available. On the other hand, the library size and simple binary programs are significant.

Evaluation of AMQP family

AMQP [14] is a wire-level protocol used for messaging. An AMQP system consists of a broker responsible for message routing between the communicating parties and a client library implementing the protocol. It does not provide any data model - only binary messages are supported. As AMQP is a broker system, implementation of request-response is cumbersome and almost two times slower than in a direct mode. Only recently, the first stable version of the protocol was released, but there is still no product that supports it. On the other hand, there are a few products using the previous, noncompliant versions of the protocol: Qpid v 0.10, OpenAMQ and RabbitMQ v 0.9, and

SwiftMQ v 0.8. The AMQP standard is still evolving and every new protocol version is not backward compatible. Moreover, the specification is still a work in progress and there is no clear indication on its future direction and support from industry [15]. Two products were evaluated, Qpid and OpenAMQ, however taking into account all the outstanding issues around AMQP, we decided to withdraw them from further investigations.

PERFORMANCE TESTS

The communication within CMW should be reliable and fast. Analysing the current usage statistics, it was estimated that the new transport over a GbE network, between a server on a new front-end (Inter Core 2 Duo, 1.5GHz, 1GB RAM, GbE) and a client running on a similar machine should handle approximately:

1) 4000msg/sec req-rep calls, payload = 4Bytes
2) 5msg/sec req-rep calls, payload = 10MBytes
3) publish 400 x 8B to 10 clients, in less than 100 msec
4) publish 30 x 8B to 10 clients, in less than 20 msec

For each candidate library all four scenarios were tested. The most interesting results were obtained from test 1 (see Figure 1), where the price for the YAMI4 dynamic model and additional hop through Qpid broker can be seen. A similar problem with IceStorm is revealed by test 3 (see Figure 2). That test also unveiled the brilliant, automatic message batching implemented in ZeroMQ.

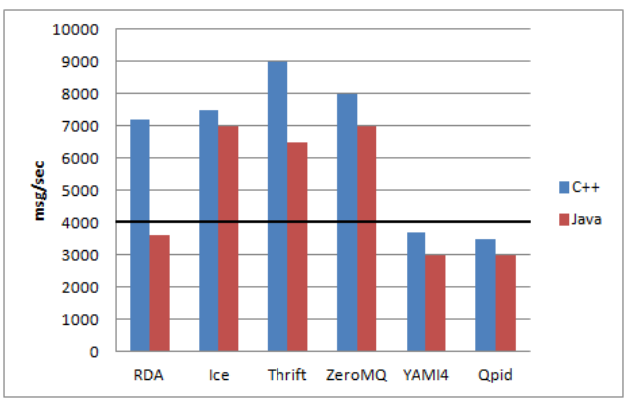


Figure 1: Test 1, a client talking to a C++ server.

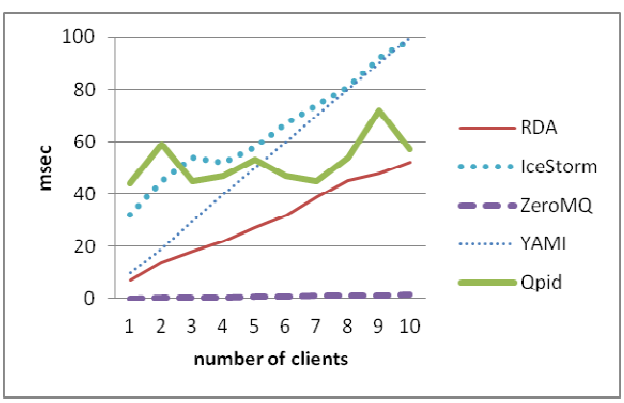


Figure 2: Test 3, pub-sub to a C++ server.

CONCLUSIONS

The paper presented several market recognized middleware products, evaluated according to the requirements of the CERN accelerator control system, as well as considering the product maturity and ease of use.

	patterns	QoS	resources	performance	user friendly	community	score
CORBA	✗	✓	✗	✓	✗	✗	2
Ice	✓	✓	✗	✓	✓	✓	5
Thrift	✗	✗	✓	✓	✗	✗	2
ZeroMQ	✓	✓	✓	✓	✓	✓	6
YAMI4	✓	✓	✓	✗	✓	✗	4
RTI	✗	✓	✗	✓	✗	✓	3
Qpid	✗	✓	✗	✗	✓	✓	3

Figure 3: Summary of evaluated middleware products.

The results are gathered in Figure 3. Three libraries were qualified for further prototyping: Ice, ZeroMQ and YAMI4. Based on prototyping the CMW team will select and adopt one of them for the future version of RDA.

REFERENCES

[1] N. Trofimov *et al.*, "Remote Device Access in the new CERN Accelerator Controls middleware", ICALEPCS 2001, San Jose, California, 2001.
[2] Z. Zaharieva *et al.*, "Database Foundation for the Configuration Management of the CERN Accelerator Controls System", ICALEPCS'11, Grenoble, France, October 2011.
[3] V. Bagiollini *et al.*, "CERN PS/SL Middleware Project, User Requirements Document", CERN Note SL/99-16(CO), Issue 1 Revision 3, Geneva, Switzerland, August 1999.
[4] OMG CORBA http://www.corba.org/
[5] M. Henning, "The rise and fall of CORBA", http://queue.acm.org/detail.cfm?id=1142044, 2006.
[6] ZeroC Ice: http://www.zeroc.com/
[7] Apache Thrift: http://thrift.apache.org/
[8] iMatix ZeroMQ : http://www.zeromq.org/
[9] Inspirel YAMI4: http://www.inspirel.com/yami4/
[10] OMG DDS: http://www.omgwiki.org/dds/
[11] OpenSplice: http://www.prismtech.com/opensplice
[12] CoreDX: http://www.twinoakscomputing.com/
[13] RTI: http://www.rti.com/
[14] AMQP: http://www.amqp.org/
[15] Pieter Hintjens, "What is wrong with AMQP", http://www.imatix.com/articles:whats-wrong-with-amqp

EPICS V4 EXPANDS SUPPORT TO PHYSICS APPLICATION, DATA ACSUISITION, AND DATA ANALYSIS*

L. Dalesio, Gabriele Carcassi, Martin Richard Kraimer, Nikolay Malitsky, Guobao Shen,
Michael Davidsaver, BNL, Upton, Long Island, New York, U.S.A
Ralph Lange, Bessy, Berlin, Germany
Matej Sekoranja, Cosylab, Ljubljana, Slovenia
James Rowland, Diamond Light Source, Oxfordshire, England
Greg White, SLAC, Menlo Park, California, U.S.A.
Timo Korhonen, PSI, Villigen, Switzerland

Abstract

EPICS version 4 extends the functionality of version 3 by providing the ability to define, transport, and introspect composite data types. Version 3 provided a set of process variables and a data protocol that adequately defined scalar data along with an atomic set of attributes. While remaining backward compatible, Version 4 is able to easily expand this set with a data protocol capable of exchanging complex data types and parameterized data requests. Additionally, a group of engineers defined reference types for some applications in this environment. The goal of this work is to define a narrow interface with the minimal set of data types needed to support a distributed architecture for physics applications, data acquisition, and data analysis.

INTRODUCTION

Version 3 of EPICS [1] consisted of three key components: the narrow client interface that presented channels of various scalar data types with metadata for display and control, a robust and high performance network protocol for these data types, and a process database that allowed the definition of control logic to achieve steady state control. Many clients were developed to provide batch, SCADA, and DCS capabilities to physics research facilities. Version 3 offered little to help develop application for data acquisition, model based control, or experiment control. Many facilities developed monolithic programs with some general libraries to build toolsets to accomplish these functions. Version 4 of EPICS, features an extended data type definition and a protocol that can serialize and de-serialize these data types to extend the narrow interface to develop services above the instrumentation level to support these functions. This paper will explore the extended interface that is being presented to the services and clients to support this functionality and the middle layer services that are planned to work in this protocol to support a modular architecture to support data acquisition, experiment control, and model based control.

VERSION 3 CHANNEL TYPES

The version 3 channel types included the basic scalar data types along with metadata to make the value useful in a process control system. These meta data included a time stamp, information that allowed the channel to be displayed, alarmed, or controlled. The time stamp is the primary mechanism for identifying when the value was read. This enables us to look for correlations between various system parameters. It also included an alarm severity that identified a channel that was not valid, or in an alarm condition. The metadata also included information to inform a client how to display a value with either display limits, display ranges and engineering units for analog type values, state labels for discreet signals, and dimensions for strings or array [2]. It also included information to understand the limit of control for analog settings and discreet outputs. These types provided reasonable functionality for device integration. They provided little to no support for data acquisition, model based control, and experiment control where the channels needed to be more complex and dense. Version 4 core allows the definition of any data structure to be serialized and transported [3]. This version defines a set of general and specific normative types to support these functions. This paper will discuss this set of normative types and their application in the middle layer services that are being developed to support this new scope.

VERSION 4 MIDDLE LAYER USE CASES

When we study the physics toolkits that have been developed such as SDDS [4], Matlab Middle Layer Toolkit [5], and XAL [6], it becomes clear that several utilities are always available. There is always a description of sets of EPICS Process Variable that are predefined for lists of channels that need to be operated on or displayed. This is specified as two services in version 4: Directory Service and Gather Service. These tools also have need to acquire information from a static data store for alignment data, magnet mapping data, and other static data that is in a relational database. These sets of data fall into several categories: named value pairs, or sets of values such as coefficients. In addition, data acquisition and model based control required large vectors of data or multidimensional arrays, or images. Data over large periods of time are also useful in this environment. It is these use cases, that drive the creation of the extended set of data types that will be exposed through the version 4 PVAccess API In addition to the extension of the data types to support these functions, The version 4 PVAccess server has been connected to the

EPCIS version 3 database to serve this data on the version 4 protocol as the first set of General Normative Types: NTFloat, NTDouble, NTEnum, NT String, etc. See Figure 1.

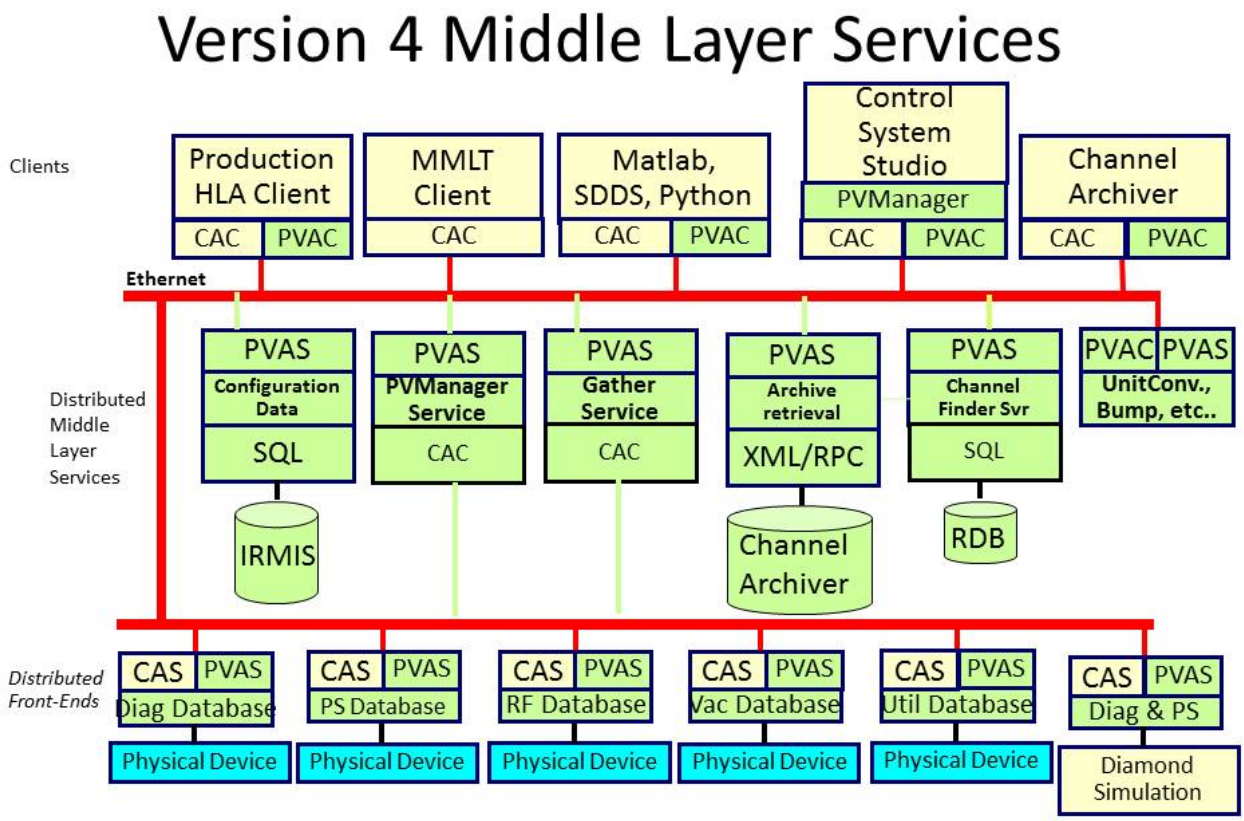


Figure 1: An EPICS Version 4 architecture drawing showing the new middle layers services that can be done with this extended set of data types. Many of these tools are being presented in this conference.

VERSION 4 NORMATIVE TYPES

Several normative data types have been defined to extend the set supported under EPICS version 3. All of the following normative types include a definition for time stamp called: NTTimeStamp , and alarm information called: NTAlarm. The general normative types are:

NTPVList, NTTimeArray, NTFreqArray, NTHistogramArray, NTMultiChannelArray, NTNDArray. NTStatisticalSamples, NTImage, and NTTable,

NTPVList

The PVList is a data structure that contains a list of process variables. In addition to the list of process variable names, this channel contains information needed to connect to this channel and provide the group as an order list. The fields may include connection information such as IP Address, port, priority, and an indication if it is currently available on the network. The field to create an ordered list could be the physical position along a beam line or storage ring.

NTTimeArray, NTFreqArray, NTHistogram

The arrays types need to be different as they represent very different data. The type informs the clients what types of information they present and the operations that can be done on them. NTTimeArray is from a digitizer and must have the time between samples to understand how it relates to other waveforms taken relative to it. NTFreqArray is an FFT of a NTFreqArray that requires the delta frequency between samples. The NTHistogram is an array of counts for different ranges of a scalar. This array can count values on either a linear distribution or log distribution..

NTMultiChannelArray

The multichannel array is a container for aggregating a set of scalar values into a single vector that is typically from the same time such as an orbit, or vacuum profile.

NTNDArray

An N dimensional array is used to transport a vector of data taken over time or a two dimensional array taken over time. This extends to an N dimensions.

NTStatisticalSample

A statistical sample is used to compress the data being sampled at a higher time frequency, into a lower frequency into a statistical sample that includes: high value, low value, mean, time of first sample, first sample, time of last sample, last sample, standard deviation,:number of samples and the or'd alarm severity of all of the samples. This data type is useful for reducing data from a longer period to a shorter period such as: raw bpm data samples at 10 KHz and reported at 1 Hz or record data being sampled at 10 Hz and being archived once every minute. It can also be used to compress a very long range of data coming back from the data archiver where the browser only has 500 pixels for displaying the data. Compression on the server side would greatly reduce the amount of network traffic and data transfer time.

NTImage

An image is a multi-dimensional array that needs additional information to know how to display it or do math operations on it. The NTImage must know the encoding and compression to be able to integrate, perform background subtraction, or display the image.

NTTable

A table is a fine way to collect a set of named parameters that are of a different type. This is a way to catch any arbitrary set of named elements that have a fixed number of values. If there is only 1 value (or row), it is a set of named value pairs. If there is more than 1 row, it is a set of named vectors. This can be used to capture a set of configuration parameters such as: experimenter name, company name, camera used, shutter speed, focus, etc… It can also be used to return the Twiss parameters for a set of magnets, where each column is one of the elements of the TWISS parameters and each row is a magnet.

CONCLUSION

PVData in Version 4 presents a set of normative types that can be aggregated, displayed, and archived. This include the version 3 normative types along with a server connected to version 3 databases. This set can be used to create general purpose servicers such as a directory service or a time synchronous vector from a distributed set of scalar values. PVAccess supports the serialization and de-serialization of PVData so that no changes are needed in this code to extend the set normative data types. More complicated services can collect the data needed from middle layer services as collections of these types and serve the results to high level applications or other middle layer services that will use them to create other results that may be served in turn to other clients or middle layer services.

This infrastructure along with this set of normative types are being used to develop the high level physics application environment for NSLS II in collaboration with the NSLS II control group, the NSLS II physics group, the SLS Control Application group and the Diamond Beam line Control Group. Early results show good performance. However, it remains to be demonstrated that this approach works and improves the overall development and maintenance of high level applications.

REFERENCES

[1] L. Dalesio, Kraimer, et. al., "EPICS Architecture", invited talk at the ICALEPCS, Tsukuba, Japan, 1991.
[2] J. Hill, et. al., "Channel Access Reference Manual", URL: http://www.aps.anl.gov/epics/base/R3-14/12-docs/CAref.html,, 2009.
[3] M. Kraimer, et. al., "pvAccess, pvData, pvIoc, pvService overview and status", EPICS Collaboration Meeting, Villigen, Switzerland, 2011.
[4] M. Borland, "SDDS Information"m URL: http://www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/SDDSInfo.shtml, 2001.
[5] G. Portmann, J. Corbett, A. Terebilo, "Middle Layer Software Manual for Accelerator Physics," LBNL Internal Report, LSAP-302, 2005. MATLAB MIDDLE LAYER TOOLKIT
[6] *XAL*, http://neutrons.ornl.gov/APGroup/appProg/xal/xal.htm

STATUS OF THE CSNS CONTROL SYSTEM

C.H.Wang, IHEP, 100049, Beijing, China

Abstract

The China Spallation Neutron Source (CSNS) is a high current proton accelerator which will officially start construction in September of 2011 in China. The CSNS control system has finished the preliminary design including devices interface determined and high level application.

This paper introduces control system design and 他和 progress of some prototypes as well as schedule and personnel plan.

INTRODUCTION

The CSNS[1] will be a 100KW accelerator based facility which is comprised of five major sections as shown in Fig. 1: a front end consisting of a 50 Kev H⁻ ion source followed by a 3MeV RFQ; a 80 MeV linac; a 1.6 GeV RCS; a 100 kw spallation neutron target which can be expanded to 500kw.

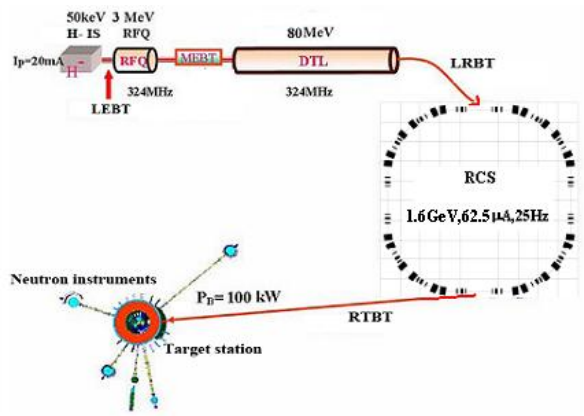


Figure 1: The Schematic Layout of the CSNS Facility.

The beam loss must be strictly controlled due to high current proton. On one hand, the radiation produced by the beam must be minimized for hands-on maintenance, on another hand, the damage to the equipments/devices due to radiation and thermal effect should be avoided since this facility has a strong radiation. So, the machine protection is very important.

CONTROL SYSTEM DESIGN

The defined scope for the control system is: that it will be a site wide monitoring and control system for the accelerator, target and conventional facilities. It will include all hardware and software of the following aspects as shown in the Fig. 2: computer system, networking, front-end controllers and hardware interface, machine protection system, as well as timing system.

The control system will not include any control or data acquisition for the target and the experimental stations. It further will not include the personnel protection system, which is a separate system that will be monitored by the control system. The control system will exchange data to conventional utility through the Ethernet. The overall task of the control system is to control and monitor thousand of the equipments/devices covering linac, LRBT, RCS and RTBT. In another word, the control system mainly provides the connections between operators/programs and accelerator equipments. Besides, it provides synchronization operation and equipments protection from the linac, RCS and the target.

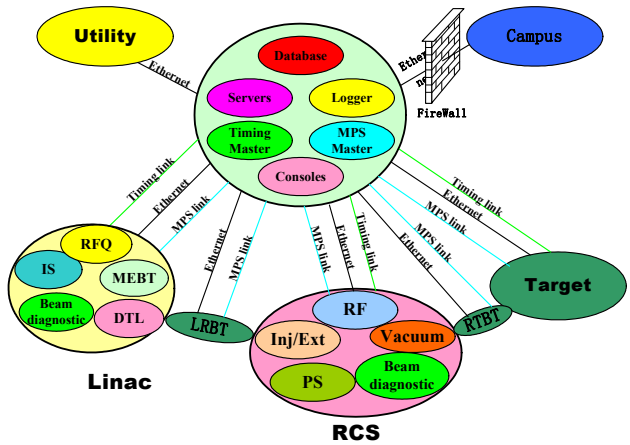


Figure 2: The Layout of the Control System Distribution.

Architecture of HW/SW

The control system will adopt the standard two layer architecture[2] with PC/Linux workstation as the Client. The EPICS IOCs are VME IOCs, one CPCI IOC, Embedded IPC IOCs and Embedded PLC IOCs for subsystem as shown in the Fig. 3. There will be no field bus to a third layer, but extensive use will be made of serial interfaces from layer two to the equipments. EPICS 3.14.10 and VxWorks 5.5/Linux 2.6 will be used for EPICS development environment. The interface from the control system to the equipment will be through different IOCs. The equipments interface has been determined.

Front End Interface

The Front-end include an ion source, LEBT,RFQ and MEBT. The ion source control system is typically a single component of the control system. It includes 5 parts: power supplies, vacuum, temperature measurement, water cooling and timing. All are required to be controlled locally in the tunnel and remotely at the central control room. The power supplies will work in the high voltage environment. So, they require high reliability and availability of the control system.

The prototype of the ion source control system was finished in 2009. All the devices had been controlled using YOKOGAWA FA-M3 PLC. One PC/Linux as EPICS IOC exchange data with the PLC CUP via Ethernet. The PLC CPU communicates with PLC I/O

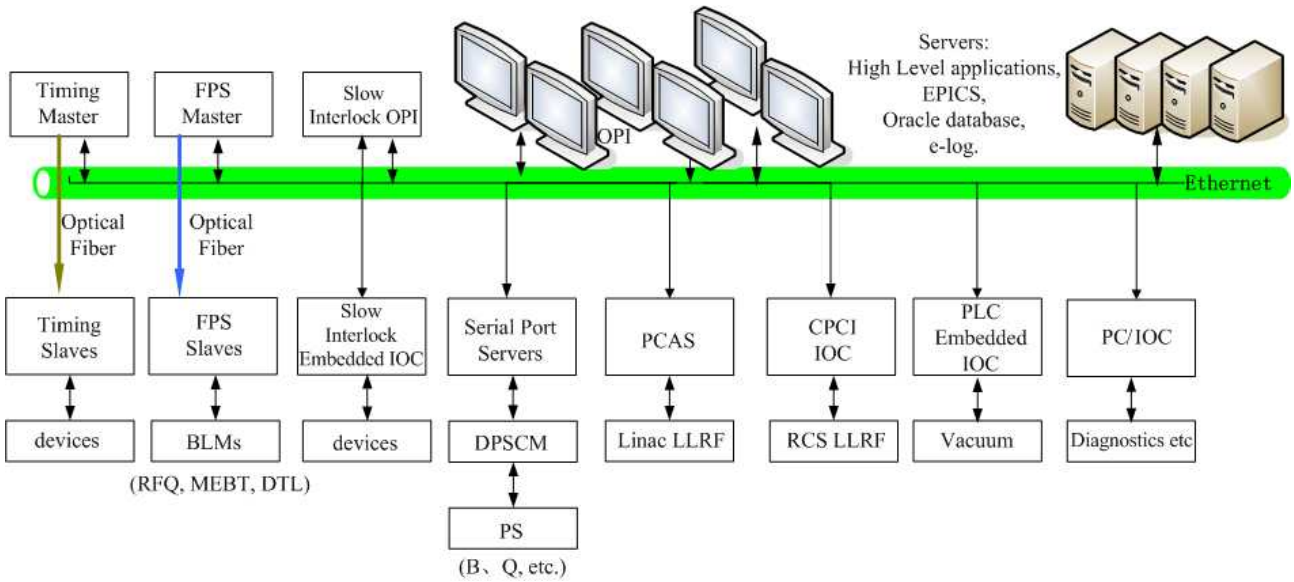


Figure 3: The Control System Architecture.

modules through the FA-Bus optical fiber. There is no extra electrical isolation between the ground and the high voltage platform. Some lessons are learn by the prototype like ground isolation, temperature modulation from relay switching on/off to PID control, more detailed specifications of the devices interface, processing of high voltage ramping and so on. In next step, the embedded IOC will be used to replace combination of the SoftIOC and the PLC/CPU as shown in the Fig. 4.

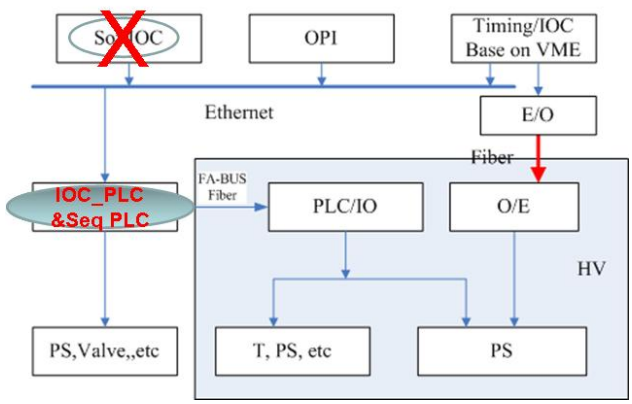


Figure 4: Ion Source Control System.

Power Supply Interface

There will be about 300 various magnet power supplies that are distributed in the linac, LRBT, RCS and RTBT. All the power supplies will be digital power supplies using a digital power supply module (DPSCM) developed by the power supply group. The DPSCM is an intelligent power supply controller with only serial port supporting Modbus RTU/RS232 and can implement logic control to the digital power supplies. Most work will be done within the DPSCM and power supplies. Since the ring is a rapid cycling synchrotron (RCS), the dipole and quadrupole power supplies with white circuit will provide 25 Hz sine waveform output to the magnets. Most magnet PS are DC power supplies, the RCS power supplies are DC plus AC power supplies with 25Hz sine waveform. Although these PS are different, the control interface to the different PS will be standardized. The requirements from the physics are very simple, for e.g. DC setpoint setting and reaback, AC amplitude/phase setting and readback. The DPSCM works fine for the RCS dipole/quadrupole power supply now.

After many times discussion between the control group and power supply control, the interface to the DPSCM has been determined using the serial port with Modbus RTU/RS232. The control group used a MOXA serial device DA682/Linux to connect the serial port of the DPSCM as shown in the Fig. 5. The communication between DA682 and the DPSCM is using Modbus RTU/RS232. We have developed Modbus RTU/RS232 driver and device support as well as database on DA682. In one word, this device is used as IOC to communicate with the DPSCM through the serial port with Modbus RTU/RS232 protocol. The prototype has been tested with the DPSCM with one power supply on site. The testing result showed that the software communication is ok and the speed can meet the requirement.

Vacuum Control Interface

The vacuum control system needs to monitor and control 128 ion pump power supply controllers, 45 vacuum gauge controllers and 27 gate valves in linac, LRBT and RCS,RTBT. The YOKOGAWA PLCs has been chosen for Ion pump and gate valve ON/OFF control and status monitoring. The MOXA serial device has been chosen for monitoring the gauges and pump power supply controllers via RS232/RS485.

#wangch@ihep.ac.cn

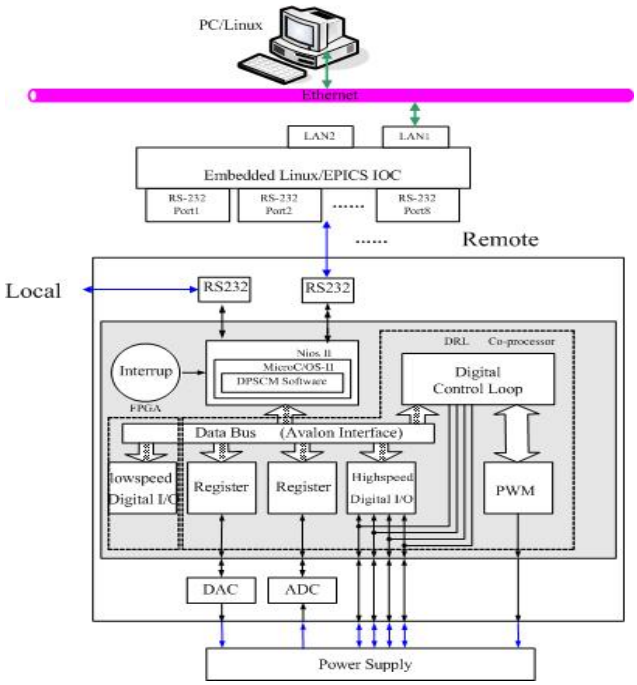


Figure 5: Power Supply Control System.

Injection and Extraction PS Interface

There are 8 injection power supplies and 8 kick pulse power supplies. YOKOGAWA WE7000 measurement has been used for the injection power supply prototype. The prototype test has been done in 12/2009. The control system is requested to monitor the PFN charge voltage and the kicker pulse current. The ZTEC oscilloscope (1GS/S, 300MHz, 12 bits, 2 Ch) with a built-in EPICS system is selected to get the charge voltage and pulse current. The ON/OFF control of the high voltage charge power supply will be implemented by YOKOGAWA PLC.

LLRF Interface

The Linac LLRF consists of a VXI crate and customized I/O module and a PC through RS422. It has been done by Linac RF group. The control system uses PCAS to communicate the API of the LLRF for changing local variable into EPICS PVs. The RCS LLRF will be implemented using customized CPCI I/O modules by RCS RF group. The control system needs to develop EPICS driver and database to interface with VxWorks driver supported by the company.

Machine Protection System

The machine protection system (MPS) has been redesigned during the preliminary stage referring to the MPS of the SNS[3] and J-Parc[4]. The design philosophy of the MPS is as follows:

- The devices local interlock done within the devices and send interlock signals to the MPS
- Hardware interlock,software recording as assitant
- Physical I/O redundancy for fatal faults in the key region/system
- Yokogawa PLC for equipment protection
- Customized Master and Slave for fast protection
- With a failsafe and rich self-diagnostic
- When a fault occur, MPS can shut down ionsource extraction high voltage and RFQ high voltage as well as perform post mortem analysis automatically

The MPS consists of two parts:

- Equipment protection for collecting signals and detecting internal faults from the equipments in Linac,RCS,Target and PPS. When any equipment fault occurs, it performs alarm handler and stop beam to avoid damage due to thermal effects
- Fast protection for interfacing BLM, once beam loss exceeds loss limit, shut down ion source and drop RFQ voltage to zero to avoid damage due to radiation and thermal effects

Besides, the MPS is also responsible for displaying status of the equipments and allowing automatic recovery from beam faults.

The equipment interlock consists of one central PLC and several PLC stations to collect signals and propagate the state signals of Linac,RCS and Target through the FL_Net to the central PLC. The FL_Net can accomplish data scanning within 200μs. The response time from a fault signal of the equipment to action can be within 20ms. How the FPS work is shown in the Fig. 6. Once the beam loss exceeds the loss limit, FPS will shut down ion source and drop RFQ voltage to zero within 20 μs to avoid damage due to radiation and thermal effects. The FPS consists of one Master and several slaves Fast responses are handled in FPGA (Xilinx Spartan 6) either locally or over bi-directional FPS optical communication. FPS unit monitors digital signals from BLMs. The ion source and RFQ are interfaced to MPS Master digital outputs. When any inputs on MPS unit changes, the output on MPS Master is triggered. The response time from MPS unit to MPS Master at 1 km distance is below 20 μs.

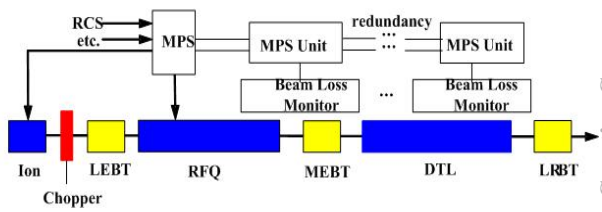


Figure 6: FPS working principle.

Timing System

Timing system is designed to provide triggers and clocks to the following systems: Front end, linac RF, injection, beam instrumentation, magnet power supply, RCS RF, extraction, spectrameter and target. The timing system consists of one EVG and several EVR stations. Detailed discussions with the above systems have been done, while there are still some points need to be studied further. Prototype using EVG/EVR has been setup. The specification of the EVG/EVR has been tested. The interface with the MPS has been studied and preliminary simulation has been done. Timing details about extraction have been studied and discussed with corresponding

systems. A dedicated hardware for the extraction timing has been designed.

Consoles

The consoles will be workstations and for cost reasons these are likely to be PCs running Linux. There are also some requirements to provide some PCs running windows for Win32 based applications. The accelerator, target and conventional facilities will be operated and monitored from the central control room, although there will be local control rooms available for device commissioning and troubleshooting.

Central Servers

There will be several central servers at the central control room. They are likely to be PC servers running Linux. They will provide development, applications, relational database, network, data archiver and viewer, alarm management, error logging, logging services and IOC booting.

Applications

The device application requirements can be met through the standard EPICS tools, control panels through EDM, alarm management through Alarm Handler, archiving through Channel Archiver together with Oracle database. Since web browsers have become an easy way to view and manipulate the control data, the CSNS control group is also planning to developed web-based control panel. Because the similarities between the CSNS and the U.S.SNS, the high level application framework will adopt XAL[5], used at SNS. XAL has been used for SNS commissioning and operation for over five years. Some XAL work in CSNS has been done including database framework of the CSNS accelerator by using standardized rules and interfaces. We have imported most of the equipments data of the CSNS Linac accelerator and two beam transfer lines. A virtual accelerator has been successfully setup. The control system will be responsible for providing high level application platform.

Network

A preliminary control system network design is based on 100Mbit switched Ethernet with a Gigabit switched Ethernet backbone. The network infrastructure will connect the control system computer to each of the local control and instrumentation areas with single and multi mode fibre. There will be a firewall between the control network and campus network. The control network will use a central core switch in the central control room and edge switches at each local control and instrumentation areas. The control network will be separate into several subnets. EPICS CA gateway will be used for the different IOC PV access and effective management of traffic and security.

CONSTRUCTION

The detailed program of work for construction phase of the project is currently being planned. The control system development will take 2 years. Some subsystem installation with the equipment installation will start in the end of 2012. The whole control system installation will be complete in 2/2016. The whole control system online commissioning will take 3 months. We are expecting to put it into operation in 5/2016. The CSNS control group is not only responsible for the CSNS control system construction, but also for the BEPCII control system maintenance. The CSNS control system would have a minimum resource requirement to get a basic system up and running.

ACKNOWLEDGEMENT

During the preliminary design stage of the control system, Prof. Songqiang Liu at SSRF in Shanghai gave many suggestion and help on some equipment interface definition. Dr. Dave Gurd at SNS who is a member of the ATAC gave me crucial advices and comments. Dr. Bob Dalesio at BNL , Mr. Ernest L. Williams Jr. at SLAC and Prof. Paul Chu at FRIB gave crucial suggestions on the control prototype and high level application. Dr. Gasper Pajor at Cosylab gave valuable suggestion on the FPS solution. The people in the control group made many contributions to the work of the control system. The author would like to thank all of them for their help and contributions.

REFERENCES

[1] CSNS design report. http://gcsns.ihep.ac.cn/
[2] http://www.aps.anl.gov/epics/
[3] C. Sibley, “Machine Protection Strategies for High Power Accelerators”, Proceedings of the 2003 Particle Accelerator Conference.
[4] J-PARC LINAC MPS Design, internal report.
[5] C.M.Chu, J.Galambos, J.Wei et al.”SNS Application Programming Plan”, ICALEPCS2001, San Jose, California, 2001

STATUS OF THE ESS CONTROL SYSTEM

Garry Trahern, European Spallation Source (ESS), Lund, Sweden

Abstract

The European Spallation Source (ESS) is a high current proton LINAC to be built in Lund, Sweden. The LINAC delivers 5 MW of power to the target at 2500 MeV, with a nominal current of 50 mA. It is designed to include the ability to upgrade the LINAC to a higher power of 7.5 MW at a fixed energy of 2500 MeV. The Accelerator Design Update (ADU) collaboration of mainly European institutions will deliver a Technical Design Report at the end of 2012. First protons are expected in 2018, and first neutrons in 2019.

The ESS will be constructed by a number of geographically dispersed institutions, which means that a considerable part of control system integration will potentially be performed off-site. To mitigate this organizational risk, significant effort will be put into standardization of hardware, software, and development procedures early in the project. We have named the main result of this standardization the Control Box concept. The ESS will use EPICS, and will build on the positive distributed development experiences of SNS and ITER.

Current state of control system design and key decisions are presented in the paper as well as immediate challenges and proposed solutions.

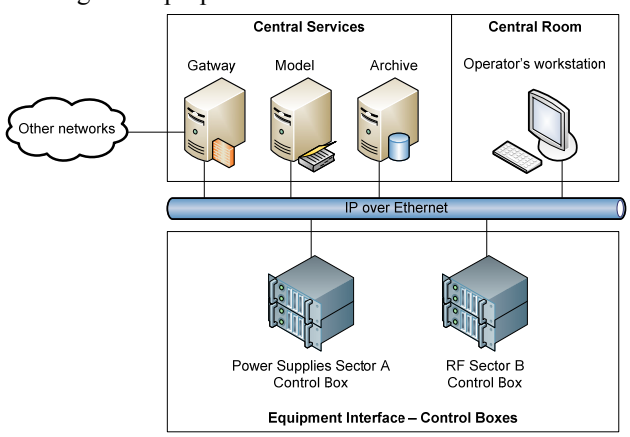


Figure 1: Three-tier architecture of the ESS control system.

CONTROL SYSTEM DESIGN

For the ESS control system, a slightly modified three-tier architecture is suitable. The three tiers [1] depicted in Fig. 1 are:

- **User interface** (upper right in the figure). These are graphical and non-graphical user interfaces. Most of them will be in the control room, but some will be elsewhere, e.g., for site-wide monitoring of the ESS status, and for remote access.
- **Central services** (upper left in the figure). Computer services that need to run continuously irrespective of user's activities, e.g., archiving of process variables' values, monitoring of alarm states, slow feedback loops, model of the machine as a whole, and management of activities that require coordination of several subsystems.
- **Equipment interfaces** (bottom in the figure). This tier is responsible for interaction with equipment and devices. It serves two purposes: to provide an abstract representation of equipment to higher layers through which the equipment can be monitored and controlled, and to implement real-time control loops.

THE CONTROL BOX

The Control Box metaphor is based on the philosophy adopted by ITER [2]. In ITER terminology the Control Box philosophy is realized with the concepts Plant System Host, CODAC, mini-CODAC and plant system I&C. It provides a standardized solution for all collaborating teams and is a cornerstone in the 3-year design update phase. An example structure of a control box is shown in Fig. 2.

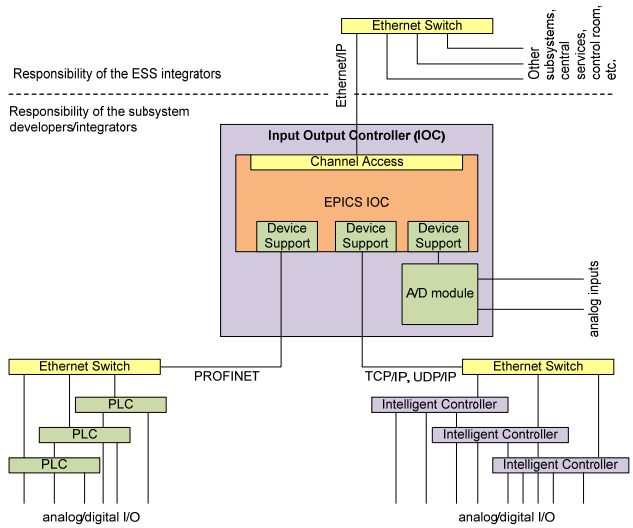


Figure 2: Schematic of Control Box components for a representative Control Box example.

Benefits

The main benefits of the Control Box are:

- To allow independent and yet standardized subsystem controls development,
- To encourage and enforce consistency between sub-systems (including target and experiments station),
- To facilitate testing of new equipment, e.g. EPICS drivers,
- To reduce risks early to prevent unexpected surprises at project integration time and
- To minimize throwaway hardware and software development.

Development

The Control Box should not be completely defined and developed too early in the project as the technology landscape to support it is rapidly evolving. We therefore expect to have iterative Control Box development.

It is necessary to develop Control Box software and hardware in (e.g., yearly) cycles. The main strategy is to start with software-only aspects (which are easiest to develop, test and distribute), and as soon as possible deliver a Control Box with a simple standardized hardware interface such as infrastructure PLC control.

In future versions functionality will be added (e.g. hardware support, including timing and feed-forward), and tools and support will evolve with available technology.

Hardware Platform

The Control Box concept enforces the usage of a common hardware platform for all teams and sub-systems to shorten the development time, support costs, etc. Therefore, the discussion on the selection of the hardware platform has been initiated early in the design stage and an iterative selection process has been agreed upon.

To further optimize the selection process a hardware selection table has been proposed where various important aspects of proposed platforms (VME, ATCA, PCI and all flavours) are leveraged against individual team requirements and scaled to the whole project size. Selection criteria for the platform includes:

- **Vendor support**: how many commercial vendors of crates and modules exist? A larger number implies that the probability of finding an off-the-shelf module for a particular task is higher.
- **Maturity**: how long has the platform been available, and how frequently is it used? Greater maturity implies lower risk and lower probability of backward-incompatible changes in the future.
- **Longevity**: how long is the platform expected to be available? Assessment should be given with the ESS's lifetime (several decades) in mind.
- **High availability**: how suitable is the platform for high-availability applications? (e.g., support for redundancy).
- **Software support**: how likely will software support (Linux driver, EPICS device support) be available for a device?
- **Prototyping vs. Production:** what is the cost of prototype development vs. production development, integration and support?
- **Risks and risk cost:** what are the risks connected to a specific platform selection and more importantly, what are the costs associated with these risks?

BEAM LINE ELEMENTS DATABASE (BLED)

ESS Control system will be built around a central configuration database (codename: BLED) where all accelerator parts and the relationships between the parts are modelled, as shown in Fig. 3.

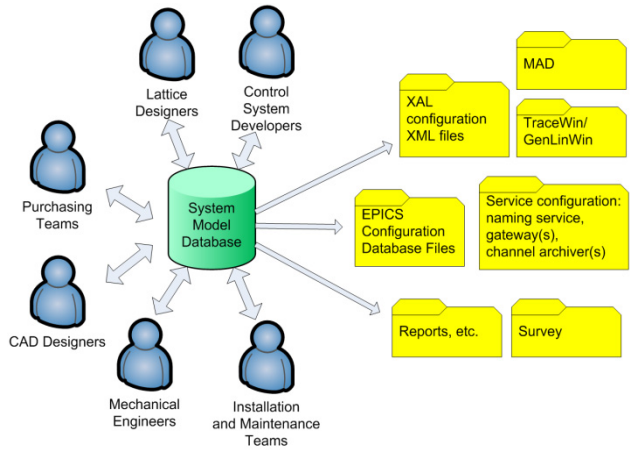


Figure 3: Users and main output of BLED.

This all-encompassing approach starts with the premise of where the equipment is located and includes data about the lattice as well as the equipment and devices. In addition, cabling information is stored in the database. The database also contains various high-level configuration parameters of the facility, all process variables, alarm definitions and alarm configurations and process variable archiving configurations. Basic information about the personnel who are making changes to the database is also included. Database entities are versioned. This makes it possible to recreate an older version of the facility model or some parts of the model, enables auditing and change tracking of the configuration and supports approval procedures for changes. The database should be constructed and operated so that it will provide a consistent model of the entire facility configuration.

Accelerator Parameters

To ensure consistency of the information being used amongst all subgroups throughout the period of accelerator design and construction, a parameter list database and web interface have been proposed in reference [3]. The main objective is to provide tools to identify inconsistencies among parameters and to enforce an ethos for groups as well as individuals to work towards a common solution. Another goal is to make the Parameter Lists a live and credible endeavour so that the data and supporting information shall be useful to a wider audience such as external reviewers as well as being easily accessible.

Control System Perspective

The use of the ESS Naming Convention in the configuration database for equipment and devices already provides the necessary information to identify all process variables in EPICS. However, in order to generate all EPICS database configuration files from BLED, a further extension of the BLED schema is needed. Therefore EPICS process variable fields, alarm server configurations and channel archiving configurations will be added to the schema to enable the generation of the EPICS database files, alarm configuration and channel archiving configuration files from the BLED configuration database. In the event that graphical database editing tools will leverage the configuration database, the schema will be extended with diagrams showing the positions of process variables on the diagrams.

DEVELOPMENT ENVIRONMENT

The Control System of the ESS will be complex (i.e., it will consist of a multitude of subsystems that will need to be integrated) and its development will require the cooperation of many developers and controls experts.

To allow proper development procedures, artefact sharing, code storage, etc., vital central services need to be set-up. Figure 4 shows a graphical overview of the environment. These services must provide the ability to:

- Share artefacts (code, documents) between engineers (Mercurial, DocDB, Plone),
- Perform version control (Mercurial),
- Continuously integrate SW development (Hudson),
- Perform complex and time consuming operations (DMSC),
- Offer a standard development environment regardless of the platform (Virtual Machine),
- Build and package software (Maven),
- Deploy artefacts to user machines or IOCs (yum, rpm),
- Perform automated testing (JUnit) and
- Report bugs and issues (Bugzilla).

Therefore, standards and guidelines must be put in place for development. Development tools and platforms should also be standardized.

ONLINE MACHINE MODEL AND SIMULATION

The user interface of the control system will most likely be based on XAL [4]. This software framework developed at SNS is a suite of Java/Python libraries and programs designed to interact with the EPICS services while providing human readable information to operate the machine. Through XAL it will be possible to change the state of the Linac by acting on all knobs and variables available to the EPICS system.

During the running of the machine, especially in the beam commissioning period, the operator has to decide which changes are needed to optimize the machine, for example to reduce losses, to steer the trajectory, etc. In this operation the user will be assisted by the Online Model: it will be a simulator of the Linac that interacts via XAL as a real machine and allows testing the parameter changes before applying to the real accelerator. Moreover the Online Model will be able to do automatic routine tasks such as loading the parameters from the running machine to find the optimal configuration for a specific problem, for example, to find the optimal configuration for correctors to steer the beam.

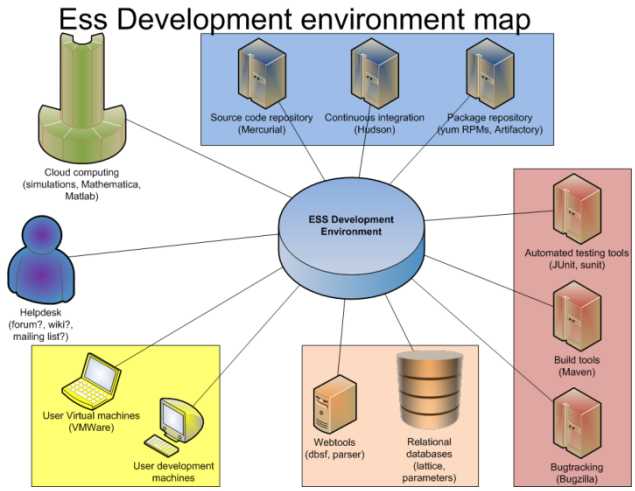


Figure 4: ESS Development environment map.

To have such instrumentality an optimal knowledge of the Linac is required, and this task can be achieved through a full campaign of simulations. Simulations are necessary to understand the dynamics of protons and the possible instabilities generated by errors in the productions of the Linac components during assembly or simply by random errors. The simulation code will be flexible enough to include unexpected phenomena arising during beam commissioning, and it will provide single and multi particle capabilities in order to predict the widest range of situations that the user can experience running the accelerator. The data to construct the simulation, e.g., systematic errors associated with component fabrication and alignment errors during installation will be maintained in the configuration database BLED.

STATUS

Design

There are three major advances in ESS controls design development that set stepping stones for further development:

- The **Conceptual Design Report** document has been drafted and describes the unified approach of controls covering the three main project segments: accelerator, target and instruments. All the development tools and services have been scaled accordingly to cover the additional segments such as

cryogenics and conventional facility infrastructure controls.

- The **Naming Convention** has been defined and approved. By learning as much as possible from ITER and SNS experiences, the naming convention was put in place early to allow proper enforcement and avoid pitfalls as the project advances.
- The initial **Costing Exercise** has been performed and provided the first estimation on the price of ESS controls. It is estimated to be less than 5% of total project cost.

Control Box

There have been several advances regarding the Control Box as well.

- The discussion on **HW platform** has been initiated with the individual teams, e.g. beam diagnostics and RF.
- It has been accepted that **early prototyping** and as-common-as-possible hardware platform is the required approach to allow the learning process to start early. This should reduce complications in sub-system integration during the construction phase of the project.
- The Control Box development cycle has been fully integrated into the **Development Environment**, see Fig. 5.

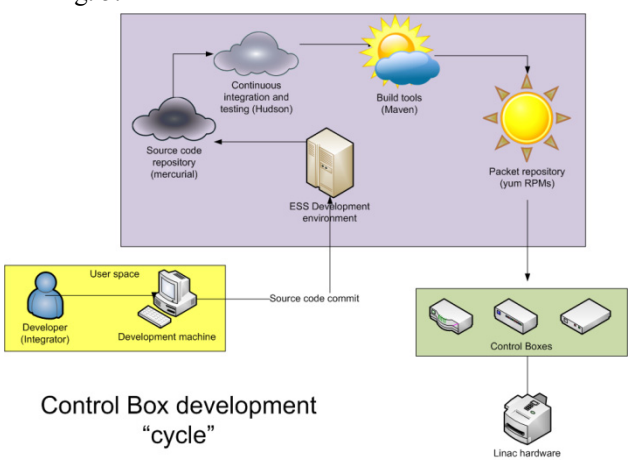


Figure 5: Control Box development cycle.

BLED

Since the Beam Line Elements Database is a vital component of the control system the development has started as well.

- The main database schema has been finalized and agreed upon.
- The Accelerator Parameters and Naming Convention tools are currently both under development.
- Various parsers and lattice tools (DBSF) have also been modified and developed to be consistent with BLED requirements.

Development Environment

The development environment as one of the key tools in the development process has been deployed in the DMSC, the Danish supercomputing cloud.

- Currently, the development environment is hosted on 8 CentOS service machines (3 designated for user's development) in the supercomputing cloud.
- It already provides an almost complete set of development tools: Mercurial, Hudson, MySQL database servers, Maven, Bugzilla, etc.
- It also provides a standalone virtual machine (based on Scientific Linux) that provides the required services for development across various platforms (Linux, Windows, Mac OS)

Timing and Machine Protection

Because of the significance of Timing and MPS to all parties (accelerator, target, instruments, safety), gathering of requirements for both have started. The possible designs and approaches have been described in the CDR documents and regular iterative meetings with different teams are underway to enforce regular collaboration and convergence for these two project-wide system.

SUMMARY

The control system is a sophisticated network connecting all the various parts of the accelerator and is essential for the synchronization and day to day running of all the equipment. Its organization will inevitably also determine its efficiency and usability. A certain degree of flexibility is expected.

Therefore, the development of the ESS control system has been from day one focused on the following key issues:

- Collaboration and communication,
- Unified development, integration and support,
- Learning from existing projects, and
- Interaction and iterative decision-making.

REFERENCES

[1] Paul D. Sheriff, Fundamentals of N-tier architecture, Barnes & Noble, 2006.
[2] F. Di Maio, et al., Development of the ITER CODAC Core Systems, Proceedings ICALEPCS2009, Kobe, Japan.
[3] K. Rathsman et al., ESS Parameter List Database and Web Interface Tools, Proceedings of IPAC2011, San Sebastian, Spain.
[4] A. Shishlo et al., The XAL Infrastructure for High Level Control Room Applications, Proceedings ICAP2009, San Francisco, CA.

STATUS OF THE ASKAP MONITORING AND CONTROL SYSTEM

J.C. Guzman, CSIRO Astronomy and Space Science, Sydney, Australia

Abstract

The Australian Square Kilometre Array Pathfinder, or ASKAP, is CSIRO's new radio telescope currently under construction at the Murchison Radio astronomy Observatory (MRO) in Mid West region of Western Australia. As well as being a world-leading telescope in its own right, ASKAP will be an important testbed for the Square Kilometre Array. This paper gives a status update of the ASKAP project and provides a detailed look at the initial deployment of the monitoring and control system as well as major issues to be addressed in future software releases before the start of system commissioning later this year.

PROJECT UPDATE

ASKAP is located in the Shire of Murchison, a remote outback mid-west region of Western Australia, approximately 400 km northeast of Geraldton and 800 km north of Perth. The location also corresponds to the Australian SKA candidate core site. This region has been identified as ideal for radio-astronomy due to its very low population density and hence a lack of man-made radio signals that would otherwise interfere with weak astronomical signals. The site is now named the Murchison Radio-astronomy Observatory (MRO).

Approximately 75% of ASKAP time will be used as a survey telescope carrying out systematic observations of the entire southern sky. The rest of the time will be allocated to targeted observations either guest projects or target of opportunity observations. For more information about ASKAP project visit [1].

At the time of writing this paper there are 9 antennas installed, 6 of them have successfully completed their Site Acceptance Test. One of them (Antenna 29) has a Single Pixel Feed (SPF) L-band receiver and is being used as a part of the Australian Very Long Baseline Interferometry (VLBI) instrument.

All Integrated Product Teams (IPT) have passed their Critical Design Review and the project is well into production mode. The first full Phase Array Feed (PAF) front-end package, back-end electronics and control software has been installed on the Parkes 12m test-bed antenna in May 2011 and has been used in interferometry mode with the Parkes 64-m antenna to measure the PAF performance as well and to be a testdbed for the monitoring and control software system of ASKAP.

The next big milestone is the installation and first light of the Boolardy Engineering Test Array (BETA) system comprising of 6 ASKAP antennas at the MRO fully equipped with PAFs, front-end and back-end electronics, and monitoring and control software. Subsequent major project milestones are described in Table 1

Table 1: ASKAP Project Milestones

Milestone	Date
MRO infrastructure completed (including central site building and power for BETA)	Jan 2012
BETA "first light" and start of commissioning	Feb 2012
Construction of all 36 antennas complete	May 2012
Pawsey centre building (host of the ASKAP supercomputer) in Perth complete	Sep 2012
MRO power plant (ASKAP)	Dec 2012
ASKAP (with PAF) complete	Dec 2013

EVOLUTION OF THE ASKAP MONITORING AND CONTROL SOFTWARE

Software Architecture Update

After the Computing Preliminary Design Review (PDR) meeting in 2009, where we presented the first design of the ASKAP Software Architecture, and subsequent discussions with System Engineering and other IPTs, we agreed to extend the definition of the TOS (as defined in [2]). The new TOS contains all the software components related to monitoring and control of the facility and provides both raw data (visibilities) and telescope meta-data for the data processing stages.

The Central Processor (CP) group contains all components related to data reduction pipelines such as calibration and imaging, including services necessary to perform those tasks such as access to sky models and RFI data. This functional decomposition allowed us also to align to the overall ASKAP Architecture, on which TOS and CP subsystems were identified. Figure 1 presents a logical view of the current overall ASKAP software architecture.

Figure 1 also shows were EPICS lives within the software architecture. The actual implementation followed a more conventional tiered design, similar to other telescopes and particle accelerators, on which there is a middle layer providing local control and interfaces to the hardware sub-

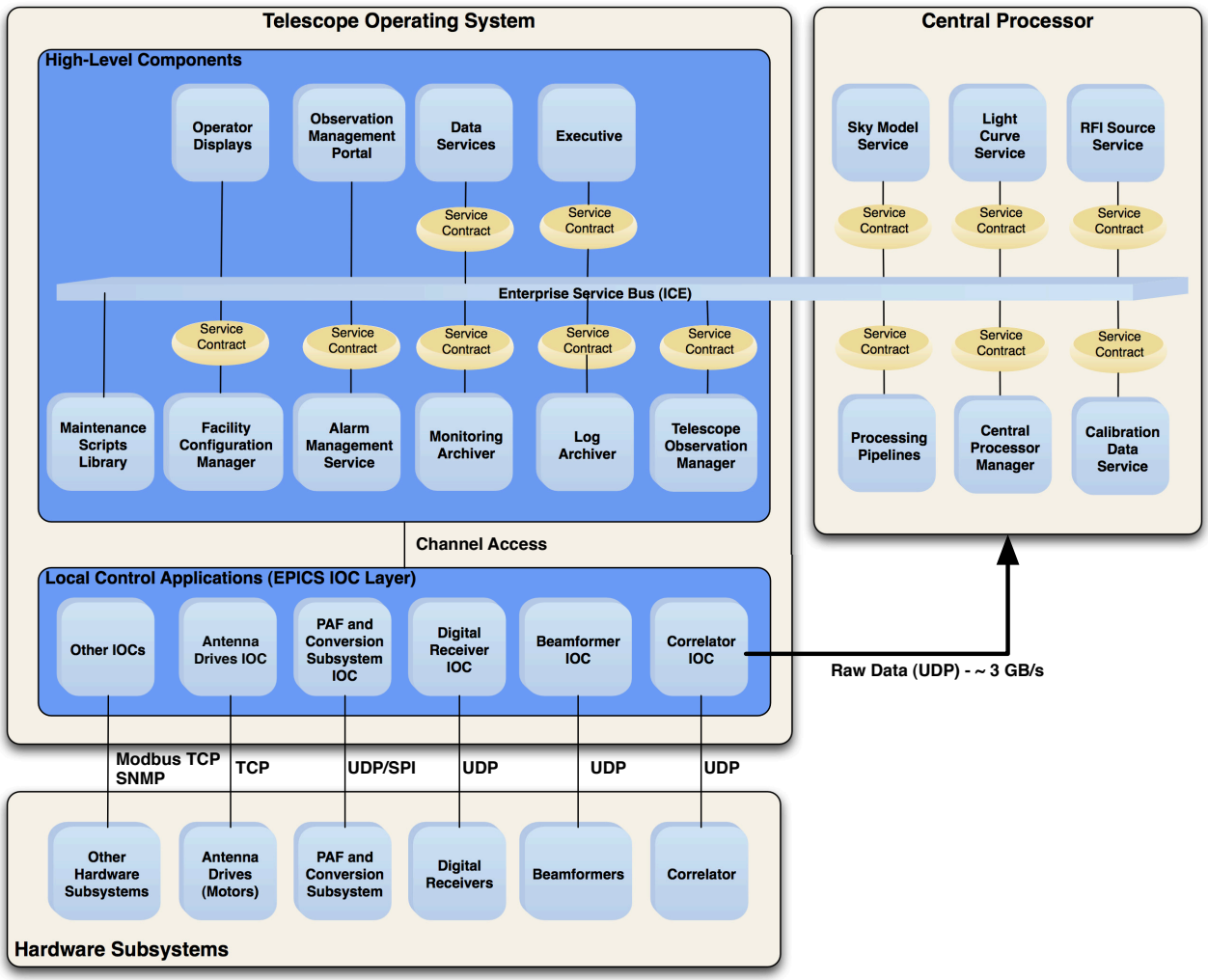


Figure 1: Logical view of the ASKAP software as implemented in latest version of TOS.

systems. This layer is implemented using EPICS IOC core. High-level components such as Monitoring Archiver, Telescope Observation Manager and Alarm Management Service use Channel Access to monitor and control the hardware subsystems. EPICS has several drivers already available for most common hardware field-bus such as SNMP and Modbus TCP. In case of our in-house UDP-based protocol (used in all our custom hardware) we have developed drivers using the EPICS C/C++ driver framework called ASYN [3].

A summary of the current control system specification is described in Table 2.

One of our requirements is to provide a flexible tool to perform maintenance operations on one or more hardware subsystems. We have developed a Python library for maintenance operations and system verification called Maintenance Script Library (MSL). This library is currently in used by engineers and instrument specialists to build specialised scripts that verifies one or more subsystems. This scripts can also be run by maintenance engineers after a piece of hardware has been replaced. The library supports concurrency and reporting. The Python library is built on top of PyEpics3 EPICS Python library developed by Matthew Neville [4].

TOS SOFTWARE RELEASES

The Build System

Since we rely heavily on several third-party software packages such as EPICS and ICE source code, we developed a powerful build system that ”wraps” several language-specific builders: scons (for C++ packages), EPICS make (for EPICS base, tools and applications), ant (for Java packages and applications), setuptools (for Python packages and applications) and autotools (for third-party packages that use autoconf). In addition, the build system supports recursive compilation, documentation generation (doxygen, Sphynx and javadocs), execution of unit and functional testing, release process via debian packages and Linux and OSX development environments.

The build system is implemented in Python and integrates also with Subversion (version control system) and

Table 2: ASKAP Monitoring and Control System Specifications

	BETA (6 Ant)	ASKAP (36 Ant)
Total Number of I/O Points	100,000	600,000
Total Number of Archived I/O Points	40,000	240,000
Monitoring data archival rate	130 GB/year	1 TB/year
Monitoring data archival rate max	1 kHz	1 kHz
Highest ”soft” control loop rate	1 Hz	1 Hz
Estimated number soft-IOC Linux computers	10	30
Science ”raw” data output rate	400 MB/s	3 GB/s

Hudson [7] (continuous integration tool).

The Development and Release Process

We have adopted an iterative and light-weight approach to our software development process, incorporating some best practices from agile methodologies. We do not use complicated Gantt Charts for our day-to-day software development planning, instead we have software release-based milestones. Every month, usually at the beginning of the month we produce a fully tested release of TOS software. Each release it is then installed and verified in the Parkes system. This monthly cycle follows the principle ”integrate early, integrate often”. This approach has been proven very useful, especially to adapt to requirement changes and external factors (hardware delay) on which we can arrange the amount of features that goes into one release or another.

Two tools has allowed us to put this methodology in practice and worth mention:

- Redmine [6], an open source project management tool that features issue tracking, milestone support, subversion integration and wiki. This tool is also being used by other ASKAP IPT and it has been a good way to do work exchange and tracking between IPTs.
- Hudson [7], an open source continuous integration tool, very easy to install, use and maintain. This tool allow us to perform continouous checking of build software as well as unit test coverage and some functional testing. Developers gets notified if they break the build.

KEY TECHNOLOGIES

EPICS

We have been using EPICS framework for the past 2 years and recently in production code in Parkes since July 2011. Overall we are very happy with the EPICS software and related tools, and more important with the openness and friendliness of the EPICS community.

Some of the EPICS tools we are currently using besides EPICS IOC core software are listed here:

- ASYN [3] for low-level EPICS driver development.
- Existing EPICS third-party drivers: devSNMP and modbus.
- PyEpics3 [4] as our Python Channel Access Client library.
- Alarm Handler (ALH). Offers a quick implementation (but limited) of the Alarm Management Service but soon to be replaced by most likely Control System Studio (CSS) BEAST.
- Sequencer and State Notation Language for state programming in the IOC applications.
- EDM for our Engineering GUIs, but starting to migrate engineering GUI development to CSS BOY.
- Control System Studio (CSS) [5]. We have just started to use CSS BOY for some of the engineering GUI and we are looking now to integrate CSS into our repository as well as some other CSS tools like BEAST, PV table, Data Browser.

ICE

We now have under our belt a good years worth of experience using ICE, and recently we have been discussing some of its short-comings (version 3.4.1). None of the short-comings are major issues and we decided that even though it may not be the most ”elegant” solution, our ICE implementation delivers do what we require in the area of high-level component communications.

Here we present some of the major issues, mainly related to ICE publish/subscribe implementation called IceStorm:

- Difficulty in debugging applications deployed in IceStorm. This is primarily due to the fact there are no observability tools which operate at the message or topic level.
- Lack of reliability features in IceStorm, particularly the inability for a subscriber to identify the fact that the broker/channel (IceStorm server) has gone away. There is no auto-reconnect feature as is present in other pub/sub middleware.
- No support for queues (only topics) so temporal decoupling is not possible.
- There is no guarantee of delivery of message. When a message is sent/published, if the subscriber is not running then the message is lost. IceStorm does not hold (or even persist) the messages and redeliver when the subscriber comes back up.

Despite these shortcomings, we have found the core functionality (Interface definition, RPC, Locator service) to be very good. Since the high-level component communication is not dominated by publish/subscribe communication these ICE issues are not critical, but we are looking into ways to improve them.

CURRENT AND FUTURE CHALLENGES

Over the past year and as we ramp up our software development one of the major challenges we encountered was how to deal with the creation of Interface Control Documentation (ICD). Since control software in our domain tends to cut across many subsystems and it is at the end of the data change, successfully delivery of fully tested software closely dependent on two things: ICD or something equivalent and the hardware to play.

An ICD is vital for developing the local control software or IOC, since it tells us with some degree how the hardware communicates and functions. We can also build emulator to assist the IOC development, integrate the IOC into the overall system for early testing and system debugging while we wait for the deliver of the full hardware.

ASKAP hardware/firmware developers responsible for production of hardware subsystems were not used (or did not have much experience) in producing ICDs. Their normal way of writing ICD and any formal documentation is to design the hardware, implement the firmware, implement some C-code software to verify the hardware and then write the ICD, by then it was too late for us and operators were forced to use the C-code program to operate the hardware while we debug and test IOC software. It took a lot of time and effort to come up with some mutual arrangement on which some form of ICD is written, even if not complete (it actually seldom is) so we can start developing, integrating and testing early into the overall system while they continue implement more functionality in the firmware.

Having an iterative approach with monthly releases have helped us greatly to adapt to changes in ICDs and late delivery of hardware.

Since we have to concentrate in the core software for start of BETA commissioning, we postponed some TOS feature to be developed over the next year or so, in particular:

- Integration of Control System Studio (CSS) into our software repository and the use of BEAST as our Alarm Management Service, with extension to capture alarms from High-level components via ICE.
- Implementation of the Facility Configuration Management.
- Implementation of the Observation Management Portal using web technologies.
- Optimisation and Refactoring of our code

We are very excited and looking forward for the upcoming months. We are hoping to get the first fringes by the end of 2011 and shortly after the first images from BETA.

REFERENCES

[1] ASKAP website `http://www.atnf.csiro.au/projects/askap`

[2] J.C. Guzman, "Preliminary Design of The Australian SKA Pathfinder (ASKAP) Telescope Control System", ICALEPCS'09, Kobe, October 2009

[3] ASYN EPICS driver framework website `http://www.aps.anl.gov/epics/modules/soft/asyn`

[4] PyEpics3 Python CA Library website `http://cars9.uchicago.edu/software/python/pyepics3`

[5] Control System Studio (CSS) website `http://cs-studio.sourceforge.net`

[6] Redmine Project Management Tool website: `http://www.redmine.org`

[7] Hudson Continuous Integration Tool website: `http://hudson-ci.org`

List of Authors

Bold papercodes indicate primary authors

— C —

— D —

— E —

— F —

— G —

— L —

— M —

— Q —

— R —

— S —

— T —

Institutes List

ACMOS INC.
Tokai-mura, Ibaraki, Japan
- Nemoto, H.

Advanced Technology Solar Telescope, National Solar Observatory
Tucson, USA
- Goodrich, B.D.
- Johansson, E.M.
- Wampler, S.B.

AGH University of Science and Technology
Krakow, Poland
- Kowalski, T.
- Mindur, B.

Albert-Ludwig Universität Freiburg
Freiburg, Germany
- Bitenc, U.X.
- Fehling-Kaschek, M.L.
- Hartert, J.
- Winkelmann, S.X.
- Zimmermann, S.

ALMA, Joint ALMA Observatory
Santiago, Chile
- Ibsen, J.P.A.
- Olguin, R.A.
- Shen, T.C.
- Soto, R.

ALTEN
Boulogne-Billancourt, France
- Rodriguez, C.

American University of Beirut
Beirut, Lebanon
- Antar, G.

ANL
Argonne, USA
- Johnson, A.N.
- Munson, F.H.
- Power, M.A.
- Rivers, M.L.
- Vasserman, I.
- Xu, J.Z.

ANSTO
Menai, Australia
- Hauser, N.
- Lam, T.K.
- Xiong, N.

Aquenos GmbH
Baden-Baden, Germany
- Marsching, S.

ASCo
Clayton, Victoria, Australia
- Corvetti, L.

Association EURATOM-CEA
St Paul Lez Durance, France
- Balorin, C.
- Caulier, G.
- Ducobu, L.
- Goniche, M.
- Jouve, M.B.
- Leroux, F.

ASsystem
St Genis Pouilly, France
- Gonzalez, J.
- Salatko-Petryszcze, C.

ASTRUM IT GmbH
Erlangen, Germany
- Schröder, H.-C.

Beckhoff Automation GmbH
Verl, Germany
- Burandt, N.
- Kuhn, J.

Bergische Universität Wuppertal
Wuppertal, Germany
- Braun, H.
- Hirschbuehl, D.
- Kersten, S.
- Kind, P.
- Lantzsch, K.
- Mättig, P.
- Zeitnitz, C.

BINP SB RAS
Novosibirsk, Russia
- Bak, P.A.
- Batrakov, A.M.
- Belov, S.
- Bolkhovityanov, D.
- Cheblakov, P.B.
- Dementyev, E.N.
- Dovzhenko, B.A.
- Fatkin, G.A.
- Galt, A.A.
- Gudkov, B.A.

- Kaplin, V.I.
- Karnaev, S.E.
- Kozak, V.R.
- Kuper, E.A.
- Logachev, P.V.
- Medvedev, L.E.
- Medvedko, A.S.
- Panov, A.
- Pavlenko, A.V.
- Sazansky, V.Ya.
- Selivanov, P.A.
- Serednyakov, S.S.
- Shubin, E.
- Sukharev, A.
- Tararyshkin, S.V.
- Tcheskidov, V.G.
- Velikanov, Y.M.
- Veremeenko, V.F.
- Vinokurov, N.
- Zaytsev, A.

BINP
Novosibirsk, Russia
- Talyshev, A.

Birmingham University
Birmingham, United Kingdom
- Martin, T.
- Thompson, P.D.

BNL
Upton, Long Island, New York, USA
- Altinbas, Z.
- Aronson, J.
- Binello, S.
- Brown, K.A.
- Campbell, I.G.
- Carcassi, G.
- Costanzo, M.R.
- D'Ottavio, T.
- Dalesio, L.R.
- Davidsaver, M.A.
- De Long, J.H.
- Eisele, W.
- Fernando, A.
- Ferreira, M.J.
- Frak, B.
- Fu, W.
- Gassner, D.M.
- Ha, K.
- Ho, C.
- Hoff, L.T.
- Hseuh, H.-C.
- Hu, Y.
- Jamilkowski, J.P.
- Kankiya, P.
- Katz, R.A.
- Kayran, D.
- Kennell, S.A.
- Kraimer, M.R.
- Lange, R.
- Laster, J.S.
- Lee, R.C.
- Louie, W.
- Malitsky, N.
- Marr, G.J.
- Marusic, A.
- Mead, J.
- Michnoff, R.J.
- Miller, T.A.
- Morris, J.
- Nemesure, S.
- Oerter, B.
- Oliveira Damazio, D.
- Olsen, R.H.
- Petkus, R.
- Piacentino, J.
- Pinayev, I.
- Poblaguev, A.
- Robert-Demolaize, G.
- Schoefer, V.
- Schoenfeld, R.F.
- Sheehy, B.
- Shen, G.
- Shroff, K.
- Singh, O.
- Tepikian, S.
- Theisen, C.
- Tian, Y.
- Vetter, K.
- Xu, H.
- Xu, W.
- Zigrosser, D.
- Zimmer, C.M.

Bologna University
Bologna, Italy
- Caforio, D.
- Sbarra, C.

CCFE
Abingdon, Oxon, United Kingdom
- Arnoux, G.
- Ben Ayed, N.
- Budd, T.
- Card, P.
- Cunningham, G.
- Devaux, S.
- Felton, R.C.
- Field, A.
- Goodyear, A.
- Harling, J.
- Khilar, S.
- Kinna, D.
- Lomas, P.J.
- McCullen, P.
- Naylor, G.A.
- Rimini, F. G.
- Stephen, A.V.

- Thomas, P.D.
- Young, I.D.
- Zastrow, K-D.

CEA/DAM/DIF
Arpajon, France
- Nicoloso, J.I.

CEA/DSM/IRAMIS/SIS2M
Gif sur Yvette, France
- Malloggi, F.
- Taché, O.

CEA/DSM/IRFU
France
- Denis, J.F.
- Gougnaud, F.
- Gournay, J.-F.
- Lussignol, Y.
- Mattei, P.
- Touzery, R.
- Uriot, D.

CEA/IRFU
Gif-sur-Yvette, France
- Gournay, J.-F.

CEA
Le Barp, France
- Arnoul, J.P.A.
- Chapuis, J.-C.
- Hurst, A.
- Manson, M.G.

CELLS-ALBA Synchrotron
Cerdanyola del Vallès, Spain
- Al-dmour, E.
- Becheri, F.
- Beltrán, D.B.
- Blanch, S.
- Camps, A.
- Costa, I.
- Coutinho, T.M.
- Cuní, G.
- Fernández-Carreiras, D.F.C.
- Gigante, J.V.
- Jamroz, J.J.
- Klora, J.
- Krause, L.
- Lidón-Simon, J.
- Martin, R.
- Matilla, O.
- Metge, J.
- Milán, A.
- Milán Otero, A.
- Moldes, J.
- Montaño, R.
- Niegowski, M.
- Pascual-Izarra, C.
- Pusó, S.
- Ranz, R.
- Reszela, Z.
- Ribas, J.
- Rubio, A.
- Rubio-Manrique, S.
- Ruz, A.
- Sanchez, O.
- Skorek, P.
- Suñé, R.

CERN
Geneva, Switzerland
- Alemany-Fernandez, R.
- Alessio, F.
- Alvarez, P.
- Andersen, M.
- Andreassen, O.O.
- Angoletta, M. E.
- Antoine, A.
- Antoniotti, F.
- Apollonio, A.
- Arfaoui, S.
- Arroyo Garcia, J.A.R.
- Arruat, M.
- Artoos, K.
- Assmann, R.W.
- Audrain, M.
- Augustinus, A.
- Ayass, M.
- Baggiolini, V.
- Barillère, R.
- Bart Pedersen, S.
- Battistin, M.
- Baudrenghien, P.
- Beckers, J.M.
- Bednarek, M.
- Bellorini, F.
- Benedikt, M.
- Bernard, F.B.
- Berry, S.
- Bestmann, P.
- Billen, R.
- Blanchard, S.
- Blanco Vinuela, E.
- Boccardi, A.
- Boccioli, M.
- Bonaccorsi, E.
- Bonneau, P.
- Botelho-Direito, J.
- Boucly, C.
- Bozyigit, S.
- Bradu, B.
- Brarda, L.
- Brightwell, M.
- Bräger, M.
- Buczak, W.
- Buffat, X.
- Burkimsher, P.C.

- Butterworth, A.C.
- Buttner, M.
- Calcoen, D.O.
- Cardoso, L.G.
- Carlier, E.
- Cattin, M.
- Charrondière, C.
- Charrue, P.
- Chebbi, M.
- Chochula, P.Ch.
- Collette, C.G.R.L.
- Copy, B.
- Corti, G.
- Csikos, D.
- De Cataldo, G.
- Delamare, Ch.
- Di Girolamo, B.
- Di Luca, S.
- Di Mauro, A.
- Dönszelmann, M.
- Donzé, M.
- Drosdal, L.N.
- Durand, Ph.
- Dworak, A.
- Ehm, F.
- Esposito, M.
- Farnham, B.
- Feniet, T.
- Fernandes, R.N.
- Fernández Adiego, B.
- Fernandez Carmona, P.
- Fortescue-Beck, E.
- Frank, M.
- Franz, S.
- Fuchsberger, K.
- Fullerton, J.
- Gabourin, S.
- Garcia Muñoz, A.
- Garnier, J.C.
- Gasior, M.
- Gaspar, C.
- Genillon, X.
- Gil Soriano, C.
- Giovannozzi, M.
- Girardot, G.
- Goddard, B.
- Godlewski, J.
- Golonka, P.
- Gomes, P.
- Gonzalez-Berges, M.
- González Cobas, J.D.
- Gorbonosov, R.
- Gourber-Pace, M.
- Gousiou, E.
- Guinchard, M.
- Gutleber, J.
- Gutzwiller, O.
- Haen, C.
- Hakulinen, T.
- Hammer, J.
- Hammouti, L.
- Hauviller, C.
- Havart, F.
- Hoibian, N.
- Holme, O.
- Holzer, E.B.
- Iglesias Gonsálvez, S.
- Izquierdo Rosas, S.
- Jackson, S.
- Jacobsson, R.
- Jacquet, D.
- Jalal, A.
- Janssens, S.M.
- Jaussi, M.
- Jensen, L.K.
- Jirdén, L.S.
- Jones, R.J.
- Jost, B.
- Juget, J.-F.
- Juteau, P.
- Kain, V.
- Khalid, O.
- Khasbulatov, D.
- King, Q.
- Klikovits, S.
- Kovalenko, S.
- Kuczerowski, J.
- Kudryavtsev, D.
- Kurepin, A.N.
- Kuzmin, A.M.
- Kwiatkowski, M.
- Ladzinski, T.
- Lamont, M.
- Lang, A.
- Le Roux, P.
- Lechman, M.
- Lehmann Miotto, G.
- Leuxe, R.
- Lewis, J.H.
- Lipiński, M.
- Losito, R.
- Lüders, S.
- Magnin, N.
- Marsili, A.
- Martin Marquez, M.
- Martin, B.
- Masi, A.
- Merezhin, A.
- Mettälä, M.
- Milcent, H.
- Misiowiec, M.
- Morón Ballester, R.
- Müller, G.J.
- Neufeld, N.
- Nguyen Xuan, J.
- Nikolaidis, F.
- Ninin, P.
- Nordt, A.
- Nunes, R.
- Nybø, M.
- Ortola Vidal, J.O.
- Page, S.T.

- Penacoba Fernandez, G.
- Pereira, M.
- Perez-Rodriguez, E.
- Peryt, M.
- Petrova, L.B.
- Pinazza, O.
- Prieto Barreiro, I.
- Raimondo, A.
- Redaelli, S.
- Reymond, H.
- Riesco, T.R.
- Rijllart, A.
- Rochez, J.
- Roderick, C.
- Roe, S.A.
- Romera, I.
- Rosinský, P.
- Sanchez-Corral Mena, E.
- Sanchez-Quesada, J.
- Sapinski, M.
- Savu, D.O.
- Scannicchio, D.
- Schindler, H.
- Schlenker, S.
- Schmidt, F.
- Schmidt, R.
- Schweitzer, P.
- Schwemmer, R.
- Semanaz, P.F.
- Serrano, J.
- Shaikh, A A.
- Shaposhnikova, E.N.
- Sigerud, K.
- Sliwinski, W.
- Sobczak, M.
- Solfaroli Camillocci, M.
- Spanggaard, J.
- Steinhagen, R.J.
- Suwalska, A.
- Svantesson, D.
- Tarasenko, P.
- Teixeira, D.D.
- Thiesen, H.
- Tilaro, F.M.
- Todd, B.
- Tonelli, G.
- Tsarouchas, C.A.
- Tückmantel, J.
- Uythoven, J.A.
- Valentini, F.
- Van der Bij, E.
- van Herwijnen, E.
- Vanga, M.
- Varela, F.
- Vestergard, H.
- Voitier, A.
- Voumard, N.
- Wagner, S.
- Wenninger, J.
- Willeman, D.
- Włostowski, T.
- Yastrebov, I.
- Zaharieva, Z.
- Zamantzas, C.
- Zerlauth, M.
- Zorin, E.
- Zwalinski, L.

CFNUL
Lisboa, Portugal
- Lopes, J.G.
- Redondo, L.M.

CIEMAT
Madrid, Spain
- Calvo, J.C.
- Cela-Ruiz, J.M.
- de la Gama, J.G.S.
- Guirao, A.
- Ibarra, A.
- Martinez Fresno, L.M.
- Molina Marinas, E.
- Moya, I.
- Pardillo, A.L.
- Salom, A.
- Sanz, S.
- Vazquez, C.

CLS
Saskatoon, Saskatchewan, Canada
- Baribeau, L.
- Beauregard, D.
- Berg, R.
- Black, G.
- Boots, M.J.
- Dolton, W.
- Hu, S.
- Hunter, D.
- Igarashi, R.
- Liu, D.
- Matias, E. D.
- Maxwell, D.G.
- Miller, C.D.
- Payne, C.G.
- Wilson, T.
- Wright, G.
- Zhang, H.

CNRS/CPT
Marseille, France
- Bousson, N.
- Hallewell, G.D.
- Mathieu, M.
- Rozanov, A.

COBIK
Solkan, Slovenia
- Bobnar, J.

- Šabjan, R.
- Žagar, K.

Columbia University
NY, USA
- Cota, E.G.

Consorzio RFX, Associazione Euratom-ENEA sulla Fusione
Padova, Italy
- Barbalace, A.
- Breda, M.
- Capobianco, R.
- Luchetta, A.
- Manduchi, G.
- Molon, F.
- Moressa, M.
- Simionato, P.
- Taliercio, C.

Cosylab
Ljubljana, Slovenia
- Dedič, J.
- Kobal, M.
- Sah, S.
- Saje, N.
- Sekoranja, M.
- Šepetavc, L.
- Štefanič, R.
- Tavčar, R.
- Žagar, A.
- Žagar, K.

CPPM
Marseille, France
- Gensolen, F.
- Hoffmann, D.
- Pisano, O.X.

CREATE
Napoli, Italy
- De Tommasi, G.
- Maviglia, F.

CSIRO ATNF
NSW, Australia
- Guzman, J.C.

Czech Republic Academy of Sciences, Institute of Physics
Prague, Czech Republic
- Nemecek, S.

Czech Technical University in Prague, Faculty of Mechanical Engineering
Prague, Czech Republic
- Doubek, M.
- Vacek, V.
- Vitek, M.

DAE/VECC
Calcutta, India
- Datta, C.
- Datta, K.
- Dutta, D.P.
- Mandi, T.K.
- Pandey, H.K.
- Sarkar, D.

DESY Zeuthen
Zeuthen, Germany
- Behera, B.
- Melkumyan, D.
- Schmidt, T.
- Wegner, P.
- Weisse, S.
- Wiesand, S.
- Winde, M.

DESY
Hamburg, Germany
- Alfaro, M.
- Ayvazyan, V.
- Duval, P.
- Flemming, M.
- Grabitz, J.
- Hinsch, K.
- Kracht, T.
- Núñez, T.
- Petrosyan, A.
- Rehlich, K.
- Rothkirch, A.
- Schlünzen, F.
- Schöps, A.
- Schütte, W.
- Sombrowski, E.
- Steffen, B.
- van der Reest, P.
- Wintersberger, E.

Diamond
Oxfordshire, United Kingdom
- Abbott, M.G.
- Chernousko, Y.S.
- Christou, C.
- Cobb, T.M.
- Coles, C.
- Gerring, M.
- Gibbons, E.P.
- Gillingham, I.J.
- Hamadyk, P.
- Heron, M.T.
- Horswell, I.
- Lay, S.C.
- Leech, M.A.
- Marchal, J.

- Mercado, R.
- Morgan, A.F.D.
- Nutter, B.J.
- Pearson, M.R.
- Pedersen, U.K.
- Price, A.G.
- Rees, N.P.
- Rehm, G.
- Rotolo, N.
- Rowland, J.
- Singleton, S.J.
- Thompson, J.A.
- Uzun, I.
- Vijayan, K.
- Wilkinson, K.G.
- Wilson, M.C.
- Woolliscroft, R.J.
- Yuan, F.

DPNC
Genève, Switzerland
- Robichaud-Veronneau, A.

DRAO
Penticton, British Columbia, Canada
- Hovey, G.J.

Durham University
Durham, United Kingdom
- Huang, B.K.
- Myers, R.M.
- Sharples, R.M.

EBG MedAustron
Wr. Neustadt, Austria
- Brett, A.B.
- Fabich, A.
- Marchhart, M.
- Moser, R.
- Thonke, M.
- Torcato de Matos, C.
- Veenstra, M.

Eckelmann AG
Wiesbaden, Germany
- Thomas, M.
- Welde, A.

Edinburgh University
Edinburgh, United Kingdom
- Wynne, B.M.

EFDA-JET
Abingdon, Oxon, United Kingdom
- Felton, R.C.
- McCullen, P.

EGO
Pisa, Italy
- Mohan, M.S.

EJIT
Hitachi, Ibaraki, Japan
- Okazaki, T.

ELETTRA
Basovizza, Italy
- Abrami, A.
- Asnicar, F.
- Battistello, L.
- Bogani, A.I.
- Borga, A.O.
- Borghes, R.
- Chenda, V.
- Cleva, S.
- Curri, A.
- De Marco, M.
- De Monte, R.
- Dos Santos, M.F.
- Ferianis, M.
- Fröhlich, L.
- Gaio, G.
- Giacuzzo, F.
- Kourousias, G.
- Lonza, M.
- Passos, G.
- Passuello, R.
- Pavlovič, L.
- Pivetta, L.
- Predonzani, M.
- Prica, M.
- Pugliese, M.
- Rossi, F.
- Scafuri, C.
- Scalamera, G.
- Strangolino, G.
- Vittor, D.
- Zambon, L.

EMBL
Heidelberg, Germany
- Brockhauser, S.

ENDIF
Ferrara, Italy
- Migliorini, N.

ENEA C.R. Frascati
Frascati (Roma), Italy
- Boncagni, L.

ESO
Garching bei Muenchen, Germany
- Andolfato, L.
- Chiozzi, G.
- Karban, R.
- Kiekebusch, M.
- Kornweibel, N.

ESRF
Grenoble, France
- Chaize, J.M.
- Claustre, L.
- Dimper, R.D.
- Duru, P.
- Epaud, F.
- Gerring, M.W.
- Götz, A.
- Goudard, O.
- Homs, A.
- Kirov, A.
- Koch, J.M.
- Meyer, J.M.
- Papillon, E.
- Petitdemange, S.
- Plouviez, E.
- Poncet, F.
- Pons, J.L.
- Regad, B.
- Scheidt, K.B.
- Susini, J.
- Svensson, O.
- Tafforeau, P.
- Taurel, E.T.
- Vedder, B.
- Verdier, P.V.

ESS Bilbao
Bilbao, Spain
- Garmendia, N.
- Muguira, L.
- Varnasseri, S.

ESS-Bilbao
Zamudio, Spain
- Arredondo, I.
- Badillo, I.
- Belver, D.
- del Campo, M.
- Echevarria, P.
- Eguiraun, M.
- Feuchtwanger, J.
- Harper, G.
- Hassanzadegan, H.

ESS
Lund, Sweden
- Trahern, G.

ETH
Zurich, Switzerland
- Di Calafiori, D.R.S.
- Dissertori, G.
- Holme, O.
- Lustermann, W.

European XFEL GmbH
Hamburg, Germany
- Karabekyan, S.
- Pannier, R.
- Pflüger, J.

EXTIA
Boulogne Billancourt, France
- Girardot, R.

F4E
Barcelona, Spain
- Cavinato, M.
- Sartori, F.

Fermilab
Batavia, USA
- Briegel, C.I.
- Carmichael, L.R.
- Church, M.D.
- Finstrom, D.
- Firebaugh, J.D.
- Hendricks, B.
- King, CA.
- McCrory, E.S.M.
- Neswold, R.
- Nicklaus, D.J.
- Patrick, J.F.
- Petrov, A.D.
- Rechenmacher, R.
- Schumann, C.L.
- Smedinghoff, J.G.
- Warner, A.
- You, J.

FZJ
Jülich, Germany
- Drochner, M.
- Fleischhauer-Fuss, L.
- Kleines, H.
- Schrader, T. E.
- Suxdorf, F.
- van Waasen, S.
- Wagener, M.

GANIL
Caen, France
- Berthe, C.
- Gillette, P.
- Haquin, C.H.
- Lechartier, M.

- Lécorché, E.
- Lemaître, E.
- Normand, G.
- Philippe, L.
- Touchard, D.T.

GA
San Diego, California, USA
- Johnson, R.D.
- Penaflor, B.G.
- Piglowski, D.A.
- Walker, M.L.

GSI
Darmstadt, Germany
- Bär, R.
- Bräuning, H.
- Fleck, T.
- Fröhlich, G.
- Haseitl, R.
- Hoffmann, T.
- Huhmann, R.
- Jülicher, S.
- Kolb, B.W.
- Kreider, M.
- Mauro, S.
- Panschow, W.
- Prados, C.
- Rauch, S.
- Schaa, V.RW.
- Schiebel, W.
- Steinmetz, S.
- Terpstra, W.W.
- Thieme, M.

Heidelberg University
Heidelberg, Germany
- Al-Shabibi, A.

HEPHY
Wien, Austria
- Wulz, C.-E.

HIA
Victoria, Canada
- Herriot, G.
- Veran, J.P.

HIT
Heidelberg, Germany
- Haberer, Th.
- Hanke, S.
- Höppner, K.
- Mosthaf, J.M.
- Peters, A.
- Stumpf, S.

Hochschule Darmstadt, University of Applied Science
Darmstadt, Germany
- Kreider, M.

Huazhong University of Science and Technology (HUST)
Wuhan, People's Republic of China
- Huang, J.
- Li, D.
- Liu, K.F.

Humboldt University Berlin, Institut für Physik
Berlin, Germany
- Oya, I.
- Schwanke, U.

HUST
Wuhan, People's Republic of China
- Hu, T.

HZB
Berlin, Germany
- Engel, D.B.
- Lange, R.
- Laux, P.
- Müller, R.

HZDR
Dresden, Germany
- Herbrand, F.
- Jainsch, R.
- Justus, M.
- Kretzschmar, N.
- Leege, K.-W.
- Michel, P.
- Schamlott, A.

I-Tech
Solkan, Slovenia
- Beltram, T.
- Juretič, T.
- Kenda, M.
- Repič, B.
- Škvarč, D.
- Valentinčič, Č.

ICER
Sofia, Bulgaria
- Valova, I.D.

ICM&MG SB RAS
Novosibirsk, Russia
- Kuchin, N.
- Lomakin, S.

ICT SB RAS
Novosibirsk, Russia
- Adakin, A.S.
- Chubarov, D.
- Nikultsev, V.

IFJ-PAN
Kraków, Poland
- Banaś, E.
- Hajduk, Z.
- Ludwin, J.
- Olszowska, J.
- Stanecka, E.

IHEP Beijing
Beijing, People's Republic of China
- Chu, Y.P.
- Ding, L.B.
- He, H.
- Hu, L.
- Jin, D.P.
- Lei, G.
- Li, J.J.
- Li, L.F.
- Liu, Y.L.
- Liu, Y.Q.
- Wang, C.H.
- Wang, L.
- Xu, G.L.
- Zhang, Y.H.
- Zhang, Y.L.
- Zhang, Z.Y.
- Zhu, K.J.
- Zhu, P.
- Zhuang, J.

IHEP Protvino
Protvino, Moscow Region, Russia
- Kopylov, L.
- Mikheev, M.S.

ILL
Grenoble, France
- Cecillon, F.
- Elaazzouzi, A.
- Jentschel, M.
- Le Goc, Y.
- Locatelli, J.
- Mary, T.
- Mutti, P.
- Ortiz, H.
- Ratel, J.
- Rey, F.
- Ruiz-Martinez, E.
- Urban, W.

IMP
Lanzhou, People's Republic of China
- An, S.
- Li, G.H.
- Li, P.
- Liu, W.F.
- Ma, L.Z.
- Mao, R.S.
- Qiao, W.M.
- Wang, Y.P.
- Wu, J.X.
- Xia, J.W.
- Yang, F.
- Yang, J.C.
- Yuan, Y.J.
- Zhang, W.
- Zhao, T.C.
- Zhou, Z.Z.

INAF-OAT
Trieste, Italy
- Cirami, R.
- Coretti, I.
- Di Marcantonio, P.

Indiana University
Bloomington, Indiana, USA
- Egorov, K.

INFN-Bari
Bari, Italy
- De Cataldo, G.
- Franco, A.
- Pastore, C.
- Sgura, I.
- Volpe, G.

INFN-Bologna
Bologna, Italy
- Bindi, M.
- Galli, D.
- Pinazza, O.
- Polini, A.

INFN-Roma II
Roma, Italy
- Ammendola, R.
- Catani, L.
- Zani, F.

INFN/LNF
Frascati (Roma), Italy
- Bisegni, C.
- Calabrò, S.
- Ciuffetti, P.
- Di Pirro, G.
- Mazzitelli, G.
- Sborzacchi, F.
- Stecchi, A.

INFN/LNL
Legnaro (PD), Italy
- Andrighetto, A.
- Bassato, G.
- Benini, D.
- Canella, S.
- Carletto, O.
- Conforto, N.
- Costa, L.
- Giacchini, M.G.
- Giovannini, L.G.
- Montano, J.A.
- Montis, M.
- Poggi, M.
- Prete, G.P.
- Vásquez, J.A.

INRIM
Turin, Italy
- Mana, G.
- Massa, E.

IPFN
Lisbon, Portugal
- Alves, D.
- Carvalho, B.
- Carvalho, I.S.
- Carvalho, P.J.
- Fernandes, H.
- Neto, A.
- Valcárcel, D.F.

IPHC
Strasbourg Cedex 2, France
- Graehling, P.G.
- Hosselet, J.H.
- Maazouzi, C.

IRAM
Saint Martin d'Heres, France
- Blanchet, S.
- Broguiere, D.
- Chavatte, P.
- Morel, F.
- Perrigouard, A.
- Torres, M.

ISEL
Lisboa, Portugal
- Lopes, J.G.

iSencia Belgium
Gent, Belgium
- De Ley, E.
- Jacobs, D.

IST
Lisboa, Portugal
- Alves, D.
- Valcárcel, D.F.

itemis
Luenen, Germany
- Völter, M.

ITEP
Moscow, Russia
- Barabin, S.V.

ITER Organization
St. Paul lez Durance, France
- Abadie, L.
- Di Maio, F.
- Evrard, B.
- Fourneron, J-M.
- Gulati, H.K.
- Hansalia, C.
- Journeaux, J.Y.
- Kim, C.S.
- Klotz, W.-D.
- Mahajan, K.
- Makijarvi, P.
- Matsumoto, Y.
- Pande, S.
- Simrock, S.
- Stepanov, D.
- Utzel, N.
- Vergara-Fernandez, A.
- Wallander, A.
- Winter, A.
- Yonekawa, I.
- Zabeo, L.

iThemba LABS
Somerset West, South Africa
- Crombie, M.A.
- Ellis, C.
- Hogan, M.E.
- Kohler, I.H.
- Mostert, H.W.
- Mvungi, M.
- Oliva, C.
- Pilcher, J.V.
- Stodart, N.

ITN
Sacavém, Portugal
- Rocha, J.

IT
Lisboa, Portugal
- Corrêa Alegria, F.A.

J-PARC, KEK & JAEA
Ibaraki-ken, Japan
- Kamikubota, N.
- Yamamoto, N.

JAEA/J-PARC
Tokai-Mura, Naka-Gun, Ibaraki-Ken, Japan
- Kato, Y.
- Kawase, M.
- Kikuzawa, N.
- Sako, H.
- Sato, K.C.
- Takahashi, H.
- Yoshikawa, H.

JAEA
Ibaraki-ken, Japan
- Ito, Y.
- Sakaki, H.

Japan Atomic Energy Agency (JAEA), International Fusion Energy Research Center (IFERC)
Rokkasho, Kamikita, Aomori, Japan
- Kojima, T.
- Narita, T.
- Nishiyama, K.
- Sakaki, H.
- Takahashi, H.
- Tsutsumi, K.

JASRI/SPring-8
Hyogo-ken, Japan
- Amselem, A.
- Fujihara, R.
- Furukawa, Y.
- Hasegawa, K.
- Hirono, T.
- Hosoda, N.
- Ishii, M.
- Ishizawa, Y.
- Joti, Y.
- Kago, M.
- Kawata, K.
- Kimura, H.
- Kiyomichi, A.
- Masuda, T.
- Matsumoto, T.
- Matsushita, T.
- Nariyama, N.
- Ohata, T.
- Saji, C.
- Sugimoto, T.
- Suzuki, S.
- Takao, M.
- Tanaka, R.
- Yamaga, M.
- Yamashita, A.

JINR
Dubna, Moscow Region, Russia
- Andreev, V.
- Gorbachev, E.V.
- Isadov, V.
- Kirichenko, A.
- Lebedev, N.I.
- Pilyar, N.V.
- Romanov, S.
- Rukoyatkina, T.V.
- Shaipov, S.
- Sorokoletov, R.
- Trubnikov, G.V.
- Volkov, V.

JLAB
Newport News, Virginia, USA
- Abbott, D.
- Allison, T.L.
- Barbosa, F.J.
- Cuevas, C.
- Dong, H.
- Gu, W.
- Jastrzembski, E.
- Joyce, M.E.
- Kaneta, S.R.
- Larrieu, T. L.
- Moffit, B.
- Nganga, N.
- Raydo, B.J.
- Slominski, C.J.
- Somov, A.
- Taylor, W.M.
- Wilson, J.
- Witherspoon, S.D.
- Yan, J.

Johannes Gutenberg University Mainz, Institut für Physik
Mainz, Germany
- Ertel, E.

JPL
Pasadena, California, USA
- Colavita, M.M.
- Dvorak, D.L.
- Ingham, M.D.
- Lindensmith, C.
- Shelton, C.
- Troy, M.
- Wagner, D.A.

JSI
Ljubljana, Slovenia
- Deliyergiyev, M.
- Mandić, I.

Kanto Information Service (KIS), Accelerator Group
Ibaraki, Japan
- Iitsuka, T.

- Motohashi, S.
- Takagi, M.
- Yoshida, S.Y.

Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
- Cerff, K.
- dos Santos Rolo, T.
- Haas, D.
- Mexner, W.
- Spangenberg, T.

Katholieke Hogeschool Sint-Lieven
Gent, Belgium
- Saey, P.

KEK/JAEA
Ibaraki-Ken, Japan
- Ishiyama, T.

KEK
Ibaraki, Japan
- Akai, K.
- Akiyama, A.
- Furukawa, K.
- Kadokura, E.
- Katoh, T.
- Kobayashi, T.
- Kurashina, M.
- Michizono, S.
- Mikawa, K.
- Miura, T.
- Miyahara, F.
- Nakagawa, H.
- Nakamura, T.T.
- Nakanishi, K.
- Odagiri, J.-I.
- Satoh, M.
- Suwada, T.
- Suzuki, T.
- Yamada, S.

Kingston University
Kingston on Thames, United Kingdom
- Hatton, L.

KIRAMS
Seoul, Republic of Korea
- Hong, S.S.
- Jung, I.S.
- Kang, K.U.
- Park, Y.S.

KIT
Eggenstein-Leopoldshafen, Germany
- Garcia, A.
- Hartmann, V.
- Huttel, E.
- Jejkal, T.
- Klein, M.
- Müller, A.-S.
- Pasic, H.
- Smale, N.J.
- Stotzka, R.
- Streit, A.
- van Wezel, J.

KU Leuven
Heverlee (Leuven), Belgium
- Deconinck, G.
- Pessemier, W.
- Raskin, G.
- Van Winckel, H.

KURRI
Osaka, Japan
- Fukutani, S.
- Hirai, Y.
- Kawabe, H.
- Kobayashi, Y.
- Kuriyama, Y.
- Miyabe, M.
- Morimoto, Y.
- Sano, T.
- Sato, N.
- Takamiya, K.
- Tanigaki, M.

LAL
Orsay, France
- Callot, O.
- Foggetta, L.G.

LANL
Los Alamos, New Mexico, USA
- Baily, S.A.
- Baros, D.
- Björklund, E.
- Hatch, C.D.
- Marroquin, P.S.
- Olivas, P.D.
- Pieck, M.
- Shelley, F.E.
- Warren, D.S.
- Winton, W.

LBNL
Berkeley, California, USA
- Byrd, J.M.
- Doolittle, L.R.
- Serrano, C.

LIP
Lisboa, Portugal
- Marques Vinagre, F.
- Ribeiro, G.
- Santos, H.F.

LLNL
Livermore, California, USA
- Adams, P.
- Azevedo, S.G.
- Beeler, R.G.
- Bettenhausen, R.C.
- Bond, E.J.
- Brunton, G.K.
- Carey, R.W.
- Casey, A.D.
- Chandrasekaran, H.C.
- Demaret, R.
- Edwards, O.D.
- Fedorov, M.A.
- Fisher, J.M.
- Fishler, B.T.
- Foxworthy, C.B.
- Frazier, T.M.
- Hutton, M.S.
- Krammen, J.E.
- Lagin, L.J.
- Liebman, J.A.
- Ludwigsen, A.P.
- Mapoles, E.A.
- Marsh, A.A.
- Marshall, C.D.
- Mauvais, J.
- Pannell, T. M.
- Parham, T.G.
- Reed, R.K.
- Sanchez, R.J.
- Sater, J.M.
- Shelton, R.T.
- Speck, D.E.
- Talbot, A.J.
- Tappero, J.D.
- Townsend, S.L.
- Van Arsdall, P.J.
- Warrick, A.L.
- Wilson, B.A.

LPI
Moscow, Russia
- Mashinistov, R.
- Zhukov, K.

MAX-lab
Lund, Sweden
- Larsson, K.
- Spruce, D.P.

MEDIANE SYSTEM
Le Pecq, France
- Noureddine, A.

MELCO SC
Tsukuba, Japan
- Fukuta, S.F.
- Kudou, T.
- Kusano, S.
- Nakamura, T.
- Takahashi, D.
- Yoshii, K.

MEPhI
Moscow, Russia
- Romaniouk, A.

Merton College
Oxford, United Kingdom
- Harrison, J.J.

Mitsubishi Electric TOKKI Systems
Amagasaki, Hyogo, Japan
- Deguchi, H.
- Hayashi, K.
- Ryoshi, M.

MPI/IPP
Garching, Germany
- Devaux, S.

MPI
Muenchen, Germany
- Barillari, T.
- Habring, J.
- Huber, J.

MSU
East Lansing, Michigan, USA
- Arabidze, G.

NASU/INR
Kiev, Ukraine
- Iakovenko, V.
- Okhrimenko, O.
- Pugatch, V.M.

National Instruments
Austin, USA
- Truchard, J.T.

National Technical University of Athens
Athens, Greece
- Iakovidis, G.
- Ikarios, E.
- Karakostas, K.
- Leontsinis, S.
- Mountricha, E.

NCP
Islamabad, Pakistan
- Hoorani, H.

NetApp, France
Paris, France
- Marie, J.F.

NFRI
Daejon, Republic of Korea
- Hahn, S.H.
- Hong, J.S.
- Kim, W.C.
- Lee, S.
- Lee, T.G.
- Lee, W.R.
- Nam, Y.U.
- Park, J.S.
- Park, M.K.
- Park, S.Y.
- Yun, S.W.

NIKHEF
Amsterdam, The Netherlands
- Boterenbrood, H.
- Hart, R.G.K.
- Verlaat, B.

NSCL
East Lansing, Michigan, USA
- Davidson, K.D.
- Davis, M.A.
- Kusler, J.
- Vincent, J.J.
- Vuppala, V.

NSRRC
Hsinchu, Taiwan
- Chang, Y.-T.
- Chen, M. L.
- Chen, J.-R.
- Chen, J.
- Chen, Y.K.
- Cheng, Y.-S.
- Chiu, P.C.
- Hsu, K.H.
- Hsu, K.T.
- Hsu, S.Y.
- Hu, K.H.
- Kuo, C.H.
- Lai, W.Y.
- Lee, D.
- Liao, C.Y.
- Liu, K.-B.
- Pan, Y.R.
- Perng, S.Y.
- Tsai, Y.L.
- Tseng, T.C.
- Wang, C.-J.
- Wang, H.S.
- Wu, C.Y.

NSU
Novosibirsk, Russia
- Kalyuzhny, V.

ORNL RAD
Oak Ridge, Tennessee, USA
- Curry, D.

ORNL
Oak Ridge, Tennessee, USA
- Chen, X.H.
- Danilova, E.
- Dickson, R.
- Hartman, S.M.
- Kasemir, K.-U.
- Purcell, J.D.
- Thompson, D.H.
- White, K.S.

Paul Scherrer Institut
5232 Villigen PSI, Switzerland
- Anicic, D.
- Armstrong, D.A.
- Bertrand, A.G.
- Bitterli, K.
- Brands, H.
- Chevtsov, P.
- Chrin, J.T.M.
- Dach, M.
- Haemmerli, F.
- Heiniger, M.
- Higgs, C.E.
- Hugentobler, W.
- Janser, G.
- Jud, G.
- Kalantari, B.
- Kapeller, R.
- Korhonen, T.
- Krempaska, R.A.
- Laznovsky, M.P.
- Lendzian, F.
- Lüdeke, A.
- Lutz, H.
- Müller, F.
- Pal, T.
- Portmann, W.
- Provenzano, M.
- Schlott, V.
- Sloan, M.C.
- Treyer, D.M.
- Vermeulen, D.
- Zimoch, E.

PNPI
Gatchina, Leningrad District, Russia
- Filimonov, V.
- Katunin, S.
- Khomutnikov, V.
- Kovalenko, S.

Private Address
Vancouver, Canada
- Gurd, D.P.

PSI
Villigen, Switzerland
- Kalantari, B.
- Korhonen, T.
- Peier, P.

RAS/INR
Moscow, Russia
- Kurepin, A.N.

RIKEN Nishina Center
Wako, Japan
- Fujimaki, M.
- Fukunishi, N.
- Komiyama, M.
- Koyama, R.

RIKEN SPring-8 Center
Sayo-cho, Sayo-gun, Hyogo, Japan
- Tanaka, H.

RIKEN Spring-8 Harima
Hyogo, Japan
- Ueno, G.

RIKEN/SPring-8
Hyogo, Japan
- Asano, Y.
- Fukui, T.
- Hara, T.
- Hatsui, T.
- Hosoda, N.
- Itoga, T.
- Maesaka, H.
- Ohshima, T.
- Otake, T.
- Otake, Y.
- Takebe, H.

RMA
Brussels, Belgium
- Jachmich, S.

SAMEER
Chennai, India
- Anitha, R.
- Balasubramanian, A.
- Mourougayane, K.

SBU
Stony Brook, New York, USA
- Grassi, V.

SCIPP
Santa Cruz, California, USA
- Mitrevski, J.

Self Employment
Private address, USA
- Eckel, B.E.

SESAME
Amman, Jordan
- Abu Ghannam, S.
- Nadji, A.
- Qazi, Z.
- Saleh, I.

SHI Accelerator Service Ltd.
Tokyo, Japan
- Uchiyama, A.

Siemens AG, Corporate Technology, CT T DE HW 4
Erlangen, Germany
- Fleck, R.

Siemens AG
Erlangen, Germany
- Back, M.
- Hagen, U.
- Heid, O.
- Hergt, M.
- Hughes, T.J.S.
- Irsigler, R.
- Kluge, T.
- Sirtl, J.

SINAP
Shanghai, People's Republic of China
- Chen, R.
- Chen, Z.H.
- Ding, J.G.
- Gu, M.
- Hu, S.M.
- Jiang, G.-Y.
- Liu, D.K.
- Shen, L.R.
- Wan, T.

- Wang, R.
- Zhao, M.H.
- Zhong, S.P.

SLAC
Menlo Park, California, USA
- Allen, W.
- Allison, S.
- Bartoldus, R.
- Boyes, M.
- Chestnut, R.P.
- Chevtsov, S.
- Cogan, J.G.
- Colocho, W.S.
- DeContreras, G.
- Dusatko, J.E.
- Fairley, D.
- Geng, Z.
- Gordon, J.B.
- Himel, T.M.
- Hoobler, S. L.
- Kim, K.H.
- Kotturi, K.D.
- Krejcik, P.
- Lahey, T.E.
- Larsen, R.S.
- Luchini, K.
- Miller, D.W.
- Natampalli, P.
- Norum, S.R.
- Olsen, J.J.
- Pandey, P.
- Peng, S.
- Piccoli, L.
- Rock, J.
- Rogind, D.
- Sass, R.C.
- Shoaee, H.
- Straumann, T.
- Strauss, E.
- Traller, R.
- White, G.R.
- Williams, E.
- Zelazny, S.
- Zhou, J.

Softwareschneiderei GmbH
Karlsruhe, Germany
- Kaiser, V.

Solaris
Krakow, Poland
- Bocchetta, C.J.
- Goryl, P.P.
- Królas, K.
- Młynarczyk, M.
- Nietubyć, R.
- Stankiewicz, M.J.
- Tracz, P.S.
- Walczak, Ł.
- Wawrzyniak, A.I.

SOLEIL
Gif-sur-Yvette, France
- Abeillé, G.
- Abiven, Y.-M.
- Betinelli-Deck, P.
- Bisou, J.
- Blache, F.
- Briquez, F.
- Buteau, A.
- Chattou, A.
- Coquet, J.
- Corruble, D.
- Elattaoui, X.
- Gourhant, P.
- Guyot, J.
- Hardion, V.H.
- Huriez, Y.
- Kewish, C.M.
- Langlois, F.
- Laulhé, C.
- Lê, S.
- Leclercq, N.
- Martinez, P.
- Medjoubi, K.
- Millet, R.
- Monteiro, P.
- Nadolski, L.S.
- Ounsy, M.
- Pierre-Joseph Zéphir, S.
- Poirier, S.
- Ravy, S.
- Renaud, G.
- Ricaud, J.P.
- Roussier, L.
- Saintin, K.S.
- Silly, M.G.
- Sirotti, F.
- Somogyi, A.
- Tournieux, A.
- Viguier, G.

Sopra Group
Aix-en-Provence, France
- Bertin, J.
- Bourguignon, G.
- Darcourt, G.

Soreq NRC
Yavne, Israel
- Berkovits, D.
- Bisyakoev, M.
- Eliyahu, I.
- Gertz, I.G.
- Grin, A.
- Halfon, S.
- Hazenshprung, N.

- Kijel, D.
- Kreisler, A.
- Mardor, I.
- Perry, A.
- Reinfeld, E.
- Silverman, I.
- Weissman, L.

SSRF
Shanghai, People's Republic of China
- Huang, G.Q.
- Lai, L.W.
- Leng, Y.B.
- Yan, Y.B.
- Yi, X.

St. Petersburg State University
St. Petersburg, Russia
- Andrianov, S.N.
- Ivanov, A.N.
- Korkhov, V.V.
- Kulabukhova, N.V.
- Lazarev, A.
- Podzyvalov, E.A.

STFC/RAL/ASTeC
Chilton, Didcot, Oxon, United Kingdom
- McMahon, S.

STFC/RAL
Chilton, Didcot, Oxon, United Kingdom
- Burge, S.R.
- Franek, B.
- Lipp, J.D.
- Nicholls, T.C.
- Phillips, P.W.

STI
Hawthorne, USA
- Thompson, P.M.

Sundance France
Matignon, France
- Veyret, J.

TCS France
Puteaux, France
- Liotard, O.

Tech-X
Boulder, Colorado, USA
- Matykiewicz, J.L.
- Pundaleeka, R.
- Shasharina, S.G.
- Wang, N.

Technion
Haifa, Israel
- Bressler, S.
- Kajomovitz, E.
- Tarem, S.

Technische Universität Darmstadt
Darmstadt, Germany
- Samman, F.A.

TEMF, TU Darmstadt
Darmstadt, Germany
- Ackermann, W.
- Franke, S.
- Weiland, T.

The Optical Sciences Company
Anaheim, California, USA
- Browne, S.

TMT
Pasadena, California, USA
- Angeli, G.Z.
- Boyer, C.
- Ellerbroek, B.L.
- Gilles, L.
- Gillies, K.K.
- MacMynowski, D.G.
- Sirota, M.J.
- Wang, L.

TRIUMF, Canada's National Laboratory for Particle and Nuclear Physics
Vancouver, Canada
- Chekulaev, S.
- Dale, D.
- Ezawa, K.
- Klassen, E.
- Lee, K.S.
- Leross, M.
- Morris, D.B.
- Mouat, M.
- Negishi, K.
- Nussbaumer, R.B.
- Pon, J.J.
- Rapaz, S.
- Richards, J.E.
- Tikhomolov, E.
- Waters, G.
- Yogendran, P.J.

Tsinghua University
Beijing, People's Republic of China
- Chen, S.
- Du, Q.
- Gong, G.H.
- Li, J.M.
- Liu, Y.

TSL
Uppsala, Sweden
- Gajewski, K.J.

TU Darmstadt
Darmstadt, Germany
- Bonnes, U.
- Burandt, C.
- Eichhorn, R.
- Enders, J.
- Hug, F.
- Klose, C.
- Konrad, M.
- Pietralla, N.
- Platz, M.

TUD
Darmstadt, Germany
- Glesner, M.
- Spies, C.
- Surapong, S.

TUM/Physik
Garching bei München, Germany
- Ostermann, A.

UCI
Irvine, California, USA
- Batraneanu, S.M.
- Chanan, G.A.
- Stancu, S.N.

UCM
Colmenarejo, Spain
- Patricio, M.A.
- Tejeda, A.

UCSC
Santa Cruz, USA
- Mast, T.S.
- Nelson, J.

UNIPD
Padova (PD), Italy
- Bertocco, M.

Universidad Politécnica de Madrid, E.T.S.I Industriales
Madrid, Spain
- Moreno, A.

Universita' degli Studi di Milano & INFN
Milano, Italy
- Tartarelli, G.F.

Universita' degli Studi di Milano e INFN
Milano, Italy
- Citterio, M.
- Meroni, C.

University of Glasgow
Glasgow, United Kingdom
- Bates, R.L.
- Bitadze, A.
- D'Auria, S.X.
- D'Auria, S.

University of Hamburg
Hamburg, Germany
- Reiswich, E.

University of Oklahoma
Norman, Oklahoma, USA
- Boyd, R.

University of Oslo
Oslo, Norway
- Sjoen, R.

University of Oxford
Oxford, United Kingdom
- Duncan, S.
- Gayadeen, S.

University of Pavia
Pavia, Italy
- Rubini, A.
- Vaga, F.

University of Pennsylvania
Philadelphia, Pennsylvania, USA
- Hance, M.
- Olivito, D.
- Wagner, P.

University of the Basque Country, Faculty of Science and Technology
Bilbao, Spain
- Etxebarria, V.
- Jugo, J.

University of Tsukuba, Graduate School of Pure and Applied Sciences,
Tsukuba, Ibaraki, Japan
- Nagai, K.

University of Wisconsin-Madison
Madison, Wisconsin, USA
- Livny, M.

Università di Roma II Tor Vergata
Roma, Italy
- Aielli, G.
- Marchese, F.
- Vitelli, R.

Université Blaise Pascal
Clermont-Ferrand, France
- Lafarguette, P.

Uppsala University
Uppsala, Sweden
- Brenner, R.

USTC/NSRL
Hefei, Anhui, People's Republic of China
- Bao, X.
- Li, C.
- Li, W.
- Liu, G.
- Wang, J.G.
- Wang, L.
- Xuan, K.

UTFSM
Valparaíso, Chile
- Morales, C.

UW-Madison/PD
Madison, Wisconsin, USA
- Magrans de Abril, M.
- Zelepoukine, S.

York University
Heslington, York, United Kingdom
- Vann, R.G.L.

Participants List

ABBOTT Michael, DLS, United Kingdom
ABEILLE Gwenaelle, SOLEIL, France
ABIVEN Yves-Marie, SOLEIL, France
ACKROYD Karen, Dsoft, United Kingdom
ALEMANY-FERNANDEZ Reyes, CERN, Switzerland
ALVAREZ Elisa, PROCON SYSTEMS, Spain
ALVAREZ Pablo, CERN, Switzerland
ALVES Diogo, Instituto de Plasmas e Fusão Nuclear, Portugal
AMAND Frank, Cosylab, Slovenia
ANDERSEN Maxim, CERN, Switzerland
ANDOLFATO Luigi, ESO, Germany
ANDREASSEN Odd Øyvind, CERN, Switzerland
ANDREEV Vasiliy, JINR, Russia
ANDRIANOV Serge, St. Petersburg State University, Russia
ANGIOLI Enrico, Vitrociset, Italy
ANTOINE Alain, CERN, Switzerland
ANTONIOTTI Fabien, CERN, Switzerland
ARNOUL Jean Paul, CEA - CESTA, France
AUDRAIN Maxime, CERN, Switzerland
AUGUSTINUS Andre, CERN, Switzerland
AYASS Myriam, CERN, Switzerland
AZEVEDO Stephen, LLNL, USA
BADILLO Inari, ESS-Bilbao, Spain
BAER Ralph, GSI, Germany
BAGGIOLINI Vito, CERN, Switzerland
BALZER Andreas, HZB, Germany
BANERIAN Stefani, University of Washington, USA
BARABIN Sergey, ITEP, Russia
BARBALACE Antonio, Consorzio RFX - EURATOM/ENEA, Italy
BARILLERE Renaud, CERN, Switzerland
BART PEDERSEN Stephane, CERN, Switzerland
BASSATO Giorgio, INFN, Italy
BATTISTELLA Andrea, INFN, Italy
BAU Jean-Claude, CERN, Switzerland
BAUCOUR Philippe, National Instruments, France
BAUVIR Christophe, IBA, Belgium
BAUVIR Bertrand, ESO, Germany
BECHERI Fulvio, CELLS - ALBA, Spain
BECK Dietrich, GSI, Germany
BEDNAREK Mateusz, CERN, Switzerland
BERKAEV Dmitry, BINP, Russia
BERRUYER Gilles, ESRF, France
BERRYMAN Eric, FRIB, USA
BESSON-CHAVANT Gaëtane, Office du Tourisme de Grenoble,
BETEVA Antonia, ESRF, France
BETINELLI Pascale, SOLEIL, France
BICKLEY Matthew, JLAB, USA
BILLEN Ronny, CERN, Switzerland
BISEGNI Claudio, INFN, Italy
BISOU Jerome, SOLEIL, France
BITADZE Alexander, CERN, Switzerland

BJORKLUND Eric, LANL, USA
BLANCHARD Sebastien, CERN, Switzerland
BLANCHET Sebastien, IRAM, France
BLANCO Enrique, CERN, Switzerland
BLAND Alastair, CERN, Switzerland
BOBNAR Jaka, Cosylab, Slovenia
BONACCORSI Enrico, CERN, Switzerland
BORGA Andrea, NIKHEF, Netherlands
BORGHES Roberto, ELETTRA, Italy
BORK Rolf, Caltech, USA
BORROWMAN Alastair, Observatory Sciences, United Kingdom
BOSSIER Vincent, Ion Beam Applications, Belgium
BOURDEAUDUCQ Sebastien, Milkymist, Germany
BOYER Corinne, TMT, USA
BRÄGER Matthias, CERN, Switzerland
BRARDA Loic, CERN, Switzerland
BRIEGEL Charlie, Fermilab, USA
BRITO Andreia, Prosegur, Portugal
BROGUIÈRE Dominique, IRAM, France
BROWN Kevin, BNL, USA
BURKIMSHER Paul, CERN, Switzerland
BUTEAU Alain, SOLEIL, France
CALCOEN Daniel, CERN, Switzerland
CALVO Julio, CIEMAT, Spain
CANELLA Stefania, INFN, Italy
CARDINES Nicola, CERN, Switzerland
CARDOSO Luis, CERN, Switzerland
CARMICHAEL Linden, Fermilab, USA
CARRONE Enzo, SLAC, USA
CASSADY Cindy, LLNL, USA
CATANI Luciano, INFN, Italy
CATTIN Matthieu, CERN, Switzerland
CAZALA Jérôme, SOPRA, France
CECILLON Franck, ILL, France
CHAIZE Jean-Michel, ESRF, France
CHAPUIS Jean-Claude, CEA - CESTA, France
CHARRAS Pierre, UJF, France
CHARRONDIERE Cedric, CERN, Switzerland
CHARRUE Pierre, CERN, Switzerland
CHEBLAKOV Pavel, Russian Academy of Sciences, Russia
CHENG Yung-Sen, NSRRC, Taiwan (ROC)
CHEVTSOV Pavel, PSI, Switzerland
CHIBA Kenji, HITACHI ZOSEN, United Kingdom
CHIOZZI Gianluca, ESO, Germany
CHOCHULA Peter, CERN, Switzerland
CHOLLET Simon, LLR-Ecole Polytechnique, France
CHRIN Jan, PSIe, Switzerland
CIACCINI Massimo, Vitrociset, Italy
CIRAMI Roberto, INAF, Italy
CLAUSTRE Laurent, ESRF, France
COBB Tom, DLS, United Kingdom
COGAN Joshua, SLAC, France
COPPOLA Nicola, XFEL, Germany
COPY Brice, CERN, Switzerland

CORRUBLE Dominique, SOLEIL, France
CORVETTI Lou, Australian Synchrotron, Australia
CSIKOS Donat, CERN, Switzerland
CUARTIELLES RUIZ David Joaquin, Arduino, Sweden
CUEVAS Chris, JLAB, USA
DACH Miroslaw, PSI, Switzerland
DALE Don, TRIUMF, Canada
DALESIO Leo, BNL, USA
DATTA Kaushik, VECC, India
DAVIDSON Kelly, FRIB, USA
DAVIS Mark, Michigan State University, USA
DE LEY Erwin, iSencia, Belgium
DE OLIVEIRA TAVARES Daniel, LNLS, Brazil
DEBELLE Thierry, National Instruments, USA
DEIWIKS Jochen, FMB, Germany
DELAGENIERE Solange, ESRF, France
DENIS Jean-François, CEA, France
DI MAIO Franck, ITER, France
DI MARCANTONIO Paolo, INAF, Italy
DI RISIO Nicolàs Alejandro, Università di Pavia, Italy
DIMPER Rudolf, ESRF, France
DONZÉ Mathieu, CERN, Switzerland
DOOLITTLE Lawrence, LBNL, USA
DRAPER Michael, CERN, Switzerland
DROCHNER Matthias, FZJ, Germany
DROSDAL Lene, CERN, Switzerland
DUBOIS Sebastien, IBA, Belgium
DUPAS Jean-Jacques, CEA - DAM, France
DURAND Gilles, CEA, France
DUSATKO John, SLAC, USA
DUVAL Philip, DESY, Germany
DWORAK Andrzej, CERN, Switzerland
ECKEL Bruce, Mindview, USA
EGUIRAUN Mikel, ESS-Bilbao, Spain
EHM Felix, CERN, Switzerland
ELIYAHU Ilan, Soreq, Israel
EMERY Rob, University of Washington, USA
EPAUD Francis, ESRF, France
FAIRLEY Diane, SLAC, USA
FAJARDO Pablo, ESRF, France
FARNHAM Ben, CERN, Switzerland
FARVACQUE Laurent, ESRF, France
FATKIN George, Russian Academy of Sciences, Russia
FEDOROV Mikhail, LLNL, USA
FERNANDEZ Leandro, CERN, Switzerland
FERNANDEZ David, CELLS - ALBA, Spain
FERNANDEZ ADIEGO Borja, CERN, Switzerland
FERNANDEZ CARMONA Pablo, CERN, Switzerland
FOGGETTA Luca, LNF, Italy
FRAK Bartosz, BNL, USA
FRANKE Sylvain, TU Darmstadt, Germany
FRANZ Sebastien, CERN, Switzerland
FRAZIER Timothy, LLNL, USA
FULLERTON John, CERN, Switzerland

FURUKAWA Kazuro, KEK, Japan
FURUKAWA Yukito, JASRI, Japan
GABOURIN Stéphane, CERN, Switzerland
GAGEY Brigitte, SOLEIL, France
GAIO Giulio, ELETTRA, Italy
GAJEWSKI Konrad, Svedberg Lab, Sweden
GARNIER Jean-Christophe, CERN, Switzerland
GASPAR Clara, CERN, Switzerland
GAYADEEN Sandira, University of Oxford, United Kingdom
GAYET Philippe, CERN, Switzerland
GERRING Matthew, ESRF, France
GERTZ Isaac, Soreq, Israel
GESSLER Patrick, XFEL, Germany
GIACCHINI Mauro, INFN, Italy
GIBBONS Edwin, DLS, United Kingdom
GILLIES Kim, TMT, USA
GILLINGHAM Ian, DLS, United Kingdom
GIOVANNINI Loris, INFN-LNL, Italy
GIRAUD Eric, ESRF,
GOMES Paulo, CERN, Switzerland
GONG Guanghua, Tsinghua University, China
GONZÁLEZ COBAS Juan David, CERN, Switzerland
GONZALEZ-BERGES Manuel, CERN, Switzerland
GOODRICH Bret, NSO/ATST, USA
GORBACHEV Evgeny, JINR, Russia
GORYL Piotr, SOLARIS, Poland
GÖTZ Andy, ESRF, France
GOUDARD Olivier, ESRF,
GOUGNAUD Françoise, CEA - IRFU, France
GOURNAY Jean-François, CEA, France
GRAILLOT Hervé, CEA, France
GRENNERAT Vincent, UJF, France
GUIJARRO Matias, ESRF, France
GUNN Steve, University of Southampton, United Kingdom
GURD David, ITER, Canada
GUTLEBER Johannes, CERN, Switzerland
GUZMAN Juan, CSIRO, Australia
HA Kiman, BNL, USA
HAEN Christophe, CERN, Switzerland
HAFOK Heiko, MPIfR, Germany
HAKULINEN Timo, CERN, Switzerland
HAMILTON Brandon, University of Cape Town, South Africa
HAMMER Josef, CERN, Switzerland
HAQUIN Christophe, GANIL, France
HARDION Vincent, SOLEIL, France
HARTMAN Steven, ORNL, USA
HASEITL Rainer, GSI, Germany
HATTON Les, Kingston University, United Kingdom
HATZIANGELI Eugenia, CERN, Switzerland
HAUSER Nick, ANSTO, Australia
HERON Mark, DLS, United Kingdom
HERRERO Javier, HV Sistemas, Spain
HIROTA Noriaki, HITACHI ZOSEN CORPORATION, Japan
HIRSCH Pascal, , Germany

HOFF Lawrence, BNL, USA
HOFFMANN Tobias, GSI, Germany
HOLLENBECK Dick, SoftPLC, USA
HOLME Oliver, ETH, Switzerland
HOMS Alejandro, ESRF, France
HOOBLER Sonya, SLAC, USA
HÖPPNER Klaus, Heidelberg Ionenstrahl-Therapie, Germany
HROVATIN Rok, Instrumentation Technologies, Slovenia
HU Shou Ming, SINAP, China
HUANG Billy, Durham University, United Kingdom
HUANG Jiang, Huazhong University, China
HUHMANN Ralf, GSI, Germany
HUR Min Goo, KAERI, South Korea
HURST Alain, CEA - CESTA, France
IVANOV Andrey, St. Petersburg State University, Russia
IVANOV Alex, Caltech, USA
IVANOV Alexander, LIGO, USA
JACOBS Dirk, iSencia, Belgium
JACOBSEN Christian, Niels Bohr Institute, Denmark
JAMILKOWSKI James, BNL, USA
JAMROZ Jerzy, CELLS - ALBA, Spain
JANSSENS Stef, CERN, Switzerland
JAUSSI Michael, CERN, Switzerland
JEAN Benjamin, Inno³, France
JENNISON Michael, EURATOM - CCFE, United Kingdom
JEZYNSKI Tomasz, DESY, Germany
JOHANSSON Anders, Lund University, Sweden
JOHNSON James, Keck Observatory, USA
JONES Robert, CERN, Switzerland
JOONEKINDT Didier, ATOS WORLDGRID,
JOUVE Michel, CEA - IRFM SIPP, France
JUSTUS Matthias, Helmholtz-Zentrum, Germany
KAGO Masahiro, JASRI, Japan
KALANTARI Babak, PSI, Switzerland
KAMIKUBOTA Norihiko, KEK, Japan
KANETA Scott, JLAB, United States
KARABEKYAN Suren, XFEL, Germany
KARBAN Robert, ESO, Austria
KASEMIR Kay-Uwe, ORNL, USA
KASPROWICZ Grzegorz, Creotech Ltd, Poland
KENDA Matej, Instrumentation Technologies, Slovenia
KENEVEY John, Facebook, USA
KERSTEN Susanne, Universität Wuppertal, Germany
KHALID Omer, CERN, Switzerland
KIM Kukhee, SLAC, USA
KINDER Stephen, Dsoft, United Kingdom
KING Charlie, Fermilab, USA
KLEINES Harald, FZJ, Germany
KLIKOVITS Stefan, CERN, Switzerland
KLORA Jörg Klora, CELLS - ALBA, Spain
KLOTZ Wolf-Dieter, ITER, France
KLUGE Thomas, Siemens, Germany
KOCH Jean Marc, ESRF, France
KOHLER Ivan, iThemba, South Africa

KOJIMA Toshiyuki, JAEA, Japan
KOLB Burkhard, GSI, Germany
KOMIYAMA Misaki, RIKEN, Japan
KONRAD Martin, TU Darmstadt, Germany
KORHONEN Timo, PSIe, Switzerland
KRACHT Thorsten, DESY, Germany
KRAIMER Martin R, ANL Retired, USA
KREIDER Mathias, GSI, Germany
KREMPASKA Renata, PSIe, Switzerland
KROFLIC Ziga, Cosylab, Slovenia
KULABUKHOVA Natalia, St. Petersburg State University, Russia
KUREPIN Alexander, CERN, Switzerland
KWIATKOWSKI Maciej, CERN, Switzerland
LADZINSKI Tomasz, CERN, Switzerland
LAGIN Lawrence, LLNL, USA
LANGE Ralph, HZB, Germany
LARRIEU Theo, JLAB, USA
LARSEN Raymond, SLAC, USA
LAUENER Joel, CERN, Switzerland
LAY Simon, DLS, United Kingdom
LE DU Patrick, IN2P3, France
LE FLOUR Thierry, LAPP, France
LECHMAN Mateusz, CERN, Switzerland
LECLERCQ Nicolas, SOLEIL, France
LECORCHE Eric, GANIL, France
LEE Ka Sing, TRIUMF, Canada
LEE Sangil, NFRI, South Korea
LEE Taegu, NFRI, South Korea
LENG Yongbin, SINAP, China
LEROUX Fabrice, CEA, France
LEWIS Hilton, Keck Observatory, USA
LI Xiaonan, IHEP, China
LI Jianwei, Canadian Light Source, Canada
LIDON Julio, CELLS - ALBA, Spain
LIPINSKI Maciej, CERN, Switzerland
LIU Gongfa KE Xuan, USTC, China
LIVNY Miron, University of Wisconsin-Madison, USA
LOCATELLI Jerome, ILL, France
LOCCI Frank, CERN, Switzerland
LÖFGREN Johan, Lund University, Sweden
LONZA Marco, ELETTRA, Italy
LOPES José, IST, Portugal
LUDWIN Jaromir, Institute of Nuclear Physics, Poland
LUEDEKE Andreas, PSIe, Switzerland
LUEDERS Stefan, CERN, Switzerland
LUTZ Hubert, PSI, Switzerland
MACARA COUTINHO Tiago, CELLS - ALBA, Spain
MACLEAN John, APS, USA
MAGNIN Nicolas, CERN, Switzerland
MAIRE Gilles, CERN, Switzerland
MAKIJARVI Petri, ITER ORGANIZATION, France
MAKSIMENKO Anton, Australian Synchrotron Company, Australia
MALOVRH Jure, COBIK, Slovenia
MANDELLI Augusto, National Instruments, Italy

MAO Ruishi, Chinese Academy of Sciences, China
MARAWAR Ravi, National Instruments, USA
MARCHANTE Daniel, PROCON SYSTEMS, Spain
MARCHHART Markus, MedAustron, Austria
MARSCHING Sebastian, Aquenos, Germany
MARSHALL Christopher, LLNL, USA
MARSILI Aurelien, CERN, Switzerland
MARTIN Paul, Australian Synchrotron, Australia
MASUDA Takemasa, JASRI, Japan
MATIAS Elder, CLS, Canada
MATILLA Oscar, CELLS - ALBA, Spain
MAYDEW Anne Françoise, ESRF,
MCGUCKIN Theodore, Jefferson Lab, USA
MEDEIROS ROMAO Luis, SCK-CEN, Belgium
MELKUMYAN David, DESY, Germany
MENGONI Fabienne, ESRF, France
MERCADO Ronaldo, DLS, United Kingdom
MEXNER Wolfgang, KIT, Germany
MEYER Jens, ESRF, France
MEZGER Anton, PSI, Switzerland
MILANOVIC Borislav, University Frankfurt, Germany
MILNE Peter, D-TACQ Solutions, United Kingdom
MIMRAN Alain, RICOH,
MISIOWIEC Marek, CERN, Switzerland
MOHAN Martin, EGO, Italy
MOLDES Jairo, CELLS - ALBA, Spain
MOLINA MARINAS Eduardo, CIEMAT, Spain
MONTIS Maurizio, INFN, Italy
MOORE Andrew, University of Cambridge, United Kingdom
MOROZOV Ivan, Russian Academy of Sciences, Russia
MORRIS David, TRIUMF, Canada
MOSER Roland, MedAustron, Austria
MOSTHAF Jörg, Heidelberg Ionenstrahl-Therapie, Germany
MOUAT Michael, TRIUMF, Canada
MÜLLER Roland, HZB, Germany
MÜLLER Gabriel, CERN, Switzerland
MUTTI Paolo, ILL, France
NADJI Amor, SESAME, Jordan
NEMESURE Seth, BNL, USA
NEMOZ Christian, ESRF, France
NESWOLD Richard, Fermilab, USA
NETO Andre, Instituto de Plasmas e Fusão Nuclear, Portugal
NEUFELD Niko, CERN, Switzerland
NGUYEN XUAN Jeremy, CERN, Switzerland
NICKLAUS Dennis, Fermilab, USA
NICOLOSO Joël, CEA - DAM, France
NOGUES Claude, CEA - CESTA, France
NÚÑEZ PARDO DE VERA María Teresa, DESY, Germany
NUTTER Brian, DLS, United Kingdom
OHLSSON Staffan, ESRF, France
OKADA Masahiro, HITACHI ZOSEN CORPORATION, Japan
OKHRIMENKO Oleksandr, Institute for Nuclear Research, Ukraine
OLSZOWSKA Jolanta, Institute of Nuclear Physics, Poland
ORTOLA Jeronimo, CERN, Switzerland

OUNSY Majid, SOLEIL, France
OYA Igor, Humboldt University, Germany
PAGE Stephen, CERN, Switzerland
PAILLARD Jean Luc, CNRS, France
PAIS Pablo, Procon Systems, S.A., Spain
PAL Trivan, PSIe, Switzerland
PANAZOL Jean Luc, LAPP, France
PANDE Sopan, ITER ORGANIZATION, France
PANOV Alexey, Russian Academy of Sciences, Russia
PAPI Enrico, CERN, Switzerland
PARK Yeun-Soo, KIRAMS, South Korea
PARK Mikyung, NFRI, South Korea
PARK Song Yeol, NFRI, South Korea
PASIC Halil, Karlsruhe Institute of Technology, Germany
PATRICK James, Fermilab, USA
PAYNE Chris, CLS, Canada
PENG Sheng, FRIB, United States
PERA Pablo, CERN, Switzerland
PEREZ Stephane, CEA, France
PERSSON Andrea G., MAX-lab, Sweden
PESSEMIER Wim, Katholieke Universiteit Leuven, Belgium
PETKUS Robert, BNL, USA
PETROVA Lyuba, CERN, Switzerland
PHILIPPE Laurent, GANIL, France
PIECK Martin, LANL, USA
PINA Mickael, Ecole Polytechnique, France
PINAZZA Ombretta, CERN / INFN, Switzerland
PITON James, LNLS, Brazil
PIVETTA Lorenzo, ELETTRA, Italy
PLESKO Mark, Cosylab, Slovenia
PLOTNIKOVA Olga, Russian Academy of Sciences, Russia
POIRIER Stéphane, SOLEIL, France
PONCET Faranguiss, ESRF, France
PORTE Dominique, ESRF, France
POWER Maria, APS, USA
PULFORD Bill, DLS, United Kingdom
RADEVA Anastasiya, CERN, Switzerland
RAINER Schwemmer, CERN, Switzerland
RAUCH Stefan, GSI, Germany
RAVAT Sylvain, CERN, Switzerland
RAVINDRAN Murali, National Instruments, USA
REES Nick, DLS, United Kingdom
REES Simon, PSI, Switzerland
REHLICH Kay, DESY, Germany
REINFELD Eyal, SOREG, Israel
REISWICH Eugen, University of Hamburg, Germany
REKOW Jens, FMB, Germany
RENAUD Guillaume, SOLEIL, France
REPIC Borut, Instrumentation Technologies, Slovenia
REVOL Jean Luc, ESRF, France
REYMOND Hubert, CERN, Switzerland
RIBEIRO Gabriel, CERN, Switzerland
RIBERON Christian, RSAI, France
RICAUD Jean-Paul, SOLEIL, France

RICHARDS Jane, TRIUMF, Canada
RIJLLART Adriaan, CERN, Switzerland
ROBICHON Marie, ESRF, France
ROD Thomas, Niels Bohr Institute, Denmark
RODERICK Chris, CERN, Switzerland
ROGIND Deborah, SLAC, USA
ROMERA Iván, CERN, Switzerland
ROS Eduardo, Seven Solutions, Spain
ROSINSKY Peter, CERN, Switzerland
ROTHKIRCH André, DESY, Germany
ROWLAND James, DLS, United Kingdom
RUBINI Alessandro, Independent Consultant, Italy
RUIZ-MARTINEZ Emilio, ILL, France
SALEH Ibrahim, SESAME, Jordan
SAMMAN Faizal, TU Darmstadt, Germany
SANCHEZ ALVAREZ Jose-Luis, CERN, Switzerland
SANFELICI Lucas, Sirius Project/LNLS, Brazil
SARTORI Filippo, F4E, Spain
SATOH Masanori, KEK, Japan
SAUNDERS Claude, APS, USA
SAVU Dan, CERN, Switzerland
SCAFURI Claudio, ELETTRA, Italy
SCHIRMER Detlev, DELTA, Germany
SCHLENKER Stefan, CERN, Switzerland
SCHÜTTE Winfried, DESY, Germany
SCHWEITZER Pierre, CERN, Switzerland
SCOTT John, RadiantBlue Technologies, USA
SEPETAVC Luka, Cosylab, Slovenia
SEREDNYAKOV Stanislav, Russian Academy of Sciences, Russia
SERRANO Carlos, LBNL, USA
SERRANO Javier, CERN, Switzerland
SEVER Franc, ESRF, France
SGURA Irene, INFN, Italy
SHEN Liren, SINAP, China
SHEN Tzu-Chiang, ALMA, Chile
SHIRKOV Grigori, JINR, Russia
SHOAEE Hamid, SLAC, USA
SIGERUD Katarina, CERN, Switzerland
SIJBRANDIJ Bart, INCAA, Netherlands
SIMROCK Stefan, ITER, France
SIROTA Mark, TMT, USA
SLIWINSKI Wojciech, CERN, Switzerland
SOLÉ V. Armando, ESRF, France
SOLONEN Vesa, Aalto University, Finland
SOMBROWSKI Elke, DESY, Germany
SPIES Christopher, TU Darmstadt, Germany
SPRUCE Darren, MAX-lab, Sweden
STARRITT Andrew, Australian Synchrotron, Australia
STEPANOV Denis, ITER, France
STEPHEN Adam, EURATOM - CCFE, United Kingdom
STOTZKA Rainer, KIT, Germany
SUGIMOTO Takashi, JASRI, Japan
SUSINI Jean, ESRF, France
SVEDA Libor, Prague Technical University, Czech Republic

SVENSSON Olof, ESRF, France
SZOSTEK Pawel, Warsaw University of Technology, Poland
TACHÉ Olivier, CEA, France
TALIERCIO Cesare, Consorzio RFX, Associazione Euratom - ENEA, Italy
TANAKA Ryotaro, JASRI, Japan
TANIGAKI Minoru, Kyoto University, Japan
TAUREL Emmanuel, ESRF, France
TAYLOR Philip, Observatory Sciences, United Kingdom
TERPSTRA Wesley W., GSI, Germany
THOMAS Geraldine, CERN, Switzerland
THOMPSON Jonathan, DLS, United Kingdom
THORNE Keith, LIGO, USA
TILARO Filippo, CERN, Switzerland
TODD Benjamin, CERN, Switzerland
TONELLI Guido, CERN, Switzerland
TONY FARRELL Tony, AAO, Australia
TOUCHARD Dominique, GANIL, France
TRAHERN Garry, European Spallation Source, Sweden
TRUCHARD James, National Instruments, USA
TSAROUCHAS Charilaos, CERN, Switzerland
TSUBOTA Kevin, Keck Observatory, USA
UCHIYAMA Akito, SHI Accelerator Service, Japan
UZUN Isa, DLS, United Kingdom
VAGA Federico, università di Pavia, Italy
VAGOVIC Patrik, ANKA, Germany
VALOVA Ivanka, ISER, Bulgaria
VAN DEN HEEVER Lize, SKA, South Africa
VAN DER BIJ Erik, CERN, Switzerland
VANDEPLASSCHE Dirk, SCK-CEN, Belgium
VARESANO Fabio, Universita' di Torino, Italy
VASQUEZ Jesus, INFN, Italy
VERAY Jean Luc, CNRS, France
VERDIER Pascal, ESRF, France
VERSTOVšEK Igor, Cosylab, Slovenia
VIGDER Mark, NRC-CNBC, Canada
VIGUIER Gregory, SOLEIL, France
VOITIER Axel, CERN, Switzerland
VOLPE Giacomo, INFN-CERN, Italy
VOLTER Markus, Voelter.de, Germany
VUPPALA Vasu, Michigan State University, USA
WAGNER Sigrid, CERN, Switzerland
WALLANDER Anders, ITER ORGANIZATION, France
WAMPLER Steve, ATST, USA
WANG Ruiping, SINAP, China
WANG Nanbor, Tech-X Corporation, USA
WANG Huai-San, NSRRC, Taiwan (ROC)
WANG Chunhong, IHEP, China
WARNER Arden, Fermilab, USA
WEISSE Stefan, DESY, Germany
WHITE Karen, ORNL, USA
WHITE Gregory, PSI, Switzerland
WILCKE Rainer, ESRF, France
WILLEMAN David, CERN, Switzerland
WILSON Martin, DLS, United Kingdom

WLOSTOWSKI Tomasz, CERN, Switzerland
WOOLLISCROFT Richard, DLS, United Kingdom
WRIGHT Glen, Canadian Light Source, Canada
XU Guanglei, IHEP, China
XU Joseph, APS, USA
XUAN Ke, USTC, China
YAMAGA Mitsuhiro, JASRI, Japan
YAMAMOTO Noboru, KEK, Japan
YAMASHITA Akihiro, SPring-8, Japan
YAN Jianxun, JLAB, USA
YASTREBOV Ilia, CERN, Switzerland
YOGENDRAN Priscilla J, TRIUMF, Canada
YOUNGMAN Christopher, XFEL.EU, Germany
ZAGAR Anze, Cosylab, Slovenia
ZAHARIEVA Zornitsa, CERN, Switzerland
ZAYTSEV Alexandr, BINP (BINP), Russia
ZERLAUTH Markus, CERN, Switzerland
ZHANG Zhaohong, SSRF, China
ZHANG Wei, Institute of Modern Physics, China
ZHUANG Jian, IHEP, China
ZIMMERMANN Stephanie, Albert-Ludwig University, Germany
ZIMOCH Elke, PSIe, Switzerland
ZUMBRUCH Peter, GSI, Germany