Proceedings of

PCaPAC 2010

October 5-8, 2010 – Eighth International Workshop on Personal Computers and Particle Accelerators

The 8th International Workshop on Personal Computers and Particle Accelerators (PCaPAC) was held from October 5th - 8th , 2010 in the City of Saskatoon, Saskatchewan, Canada. The workshop, organized by the Canadian Light Source, attracted 115 participants from 14 countries. A total of 30 talks and 48 posters were presented at the workshop. LabView training sessions were sponsored and held by National Instruments while workshops were presented covering Control System Studio, RTEMS and Matlab. A scientific excursion brought the participants to the Canadian Light Source where five areas of the facility were visited.

A banquet was held at the Western Development Museum where participants toured Boomtown , a representation of a typical Saskatchewan town in 1910. Over 30 buildings portray community life – from the general store to the working blacksmith shop to the train station and the sod house. During the banquet, the Isamu Abe Prize was presented to two winners, Marcin Trycz of INFN and Ziga Kroflic of the University of Ljubljana. Best Poster Prizes were presented to Jutta Fitzek (GSI, Darmstadt) and Michel Fodje (CLS, Saskatoon)

Special thanks go to the International Program Committee led by Elder Matias. Valuable suggestions were made to assure the scientific success and to cover a broad spectrum of topics: Accelerator Controls, Control Hardware and Low-level Software, Data Networking and Web Technology, Experiment Data Acquisition and Analysis Software, Facility and System Engineering.

On behalf of the International Program Committee and the Local Organizing Committee we express our sincere thanks for the generous financial support received from our two main sponsors, Cosylab and National Instruments. The workshop also benefitted from the financial support of six industrial exhibitors; Cosylab, Hytec, Instrumentation Technologies, Pro-Dex, National Instruments and W-ie-ne-r

We are pleased to provide the proceedings from the workshop that contain not only the 78 paper contributions received but also pdf copies of the talks and posters presented at the workshop. These proceedings were edited and prepared for publication on the Web by the local JACoW team led by Carl Finlay.

Last but not least: The highly qualified talks, posters and numerous discussions made PCaPAC 2010 a very successful and inspiring week – a warm thank you to all participants!

Sincerely

Elder Matias
Chairman - PCaPAC 2010

Contents

International Program Committee

Reinhard Bacher - BESSY (Germany)
Ralph Baer - GSI (Germany)
Giorgio Bassato - INFN (Italy)
Matthew Bickley - Jefferson Labs (US)
Daniele Bulfone - Elettra (Italy)
Luciano Catani - INFN (Italy)
Ron Chestnut - Stanford-SLAC (US)
Pavel Chevtsov - Jefferson Labs (US)
Matthias Clausen - DESY (Germany)
Philip Duval - DESY (Germany)
Richard Farnsworth - (AS) (Australia)
Dave Gurd - Oak Ridge (retired)
Mark Heron - Diamond (UK)
Kuo-tung Hsu - NSRRC (Taiwan)
Steve Hunt - Alceli (Switzerland)
Norihiko Kamikubota - KEK (Japan)
Ajith Kumar- NSC (India)
Shin-ichi Kurokawa - KEK (Japan)
Ralph Lange - BESSY (Germany)
Elder Matias - CLS (Canada)
Wolfgang Mexner - FZK (Germany)
Hiroshi Nishimura - KEK, (Japan)
Mark Plesko - Josef Stephan Institute (Slovenia)
Hermann Schmikler - CERN (Switzerland)
Guobao Shen - BNL (US)
Ryotara Tanaka - Spring 8 (Japan)
Jorg Klora - CELLS (Spain)
In Soo Ko - Postech (South Korea)
Deborah Quock - APS (US)
Bei Bei Shao - (China)
Detlef Vermeulen - PSI (Switzerland)
Igor Verstovsek - CozyLab (Slovenia)
Karen White - Oak Ridge-SNS (US)
Ernest Williams - SLAC Stanford (US)
Akihiro Yamashita - Spring 8 (Japan)

Local Organizing Committee (all Canadian Light Source)

Chairman - Elder Matias
Editor and Program - Carl Finlay
Workshop Coordinator - Dionne Laprairie
Tour Coordinator - Gillian Black
Audio/Video Coordinator -Robby Tabber
Marketing Coordinator - Sandra Ribeiro
Technical Program - Glen Wright

PCaPAC 2010 Participants

Isamu Abe Prize Winner (Shared) – Ziga Kroflic (University of Ljubljana, Ljubljana, Slovenia)

Isamu Abe Prize Winner (Shared) – Marcin Ireneusz Trycz (INFN- Roma II, Rome, Italy)

Best Wednesday Poster Winner – Jutta Fitzek (GSI – Darmstadt, Germany)

Best Thursday Poster Winner – Michel Fodje (Canadian Light Source, Saskatoon, Canada)

Accepted by Russ Berg, CLSI


Exercise: Taking a Basic Measurement
• Acquire temperature signal
• Write to graph


PCaPAC 2010
Databrowser Property view


ESS Controls Landscape
12+ (not-so-blind) institutions, 6-7 years: one elephant
Some fundamental tensions:
Evolving specifications and technology => later is better!
Late (control) system integration => earlier is better!
Enforce standards across project => centr better!
Encourage development, buy-in => distri tter!
Organizational (social) vs technical risks
We're attempting to learn from the (mis)adver
SNS... ITER... You (yes, you!)


Multi-Axis Motion Control
• 1 to 10 axes per controller
• Absolute encoders
• Ethernet, USB, RS232
• VME, PCI, PC104 bus
Multi-Axis Motion Control
Custom Solutions
PCIx Motion Controller
MAXnet Motion Controller
PC78 Motion Controller
Accessories
UMX Motion Controller


NATIONAL INSTRUMENTS
NI LabVIEW for Big Physics
Single-software platform for data acquisition, analysis, and control
Native support for real-time, parallel, and custom FPGA-based programming
Open, industry standard, COTS hardware platform – PXI/CompactPCI, NI CompactRIO
Connectivity through EPICS, OPC, Modbus, IEEE 1588 and EtherCAT
ni.com/physics
LabVIEW
ni.com


WIENER
A Phoenix Mecano Company
Data Acquisition & Control
VME64x-VME-PCI-USB
Industry Packs-CAM
ADCs-DACs-Scale
Digitizers-Steppe
Networked IOC Co
Software and Sy
Universal Multi-channel LV/HV Power Supply System
Mpod
12 Multi-Channel Floating LV System
9010 IOC
Input Output Controller
ATOM PC104 Input Output Controller using Industry Packs
Complete EPICS IOC
solution using RTEMS.
HYTEC ELECTRO
NOW HAVE
EPICS
ASYNDRIVER


we make accelerators work
How many of your control system projects did you really finish on time and on budget?
How many had the scope, time and budget defined at all?
How often did you write up the documentation?
Over 30 labs use Cosylab, the specialized control system integrator for accelerators
cosylab
CONTROL SYSTEM LABORATORY
and radio telescopes

PRESENTATION ONLY

PRESENTATION ONLY

PRESENTATION ONLY

USE OF THE CELL ACCELERATOR PLATFORM FOR SYNCHROTRON DATA ANALYSIS

J. Qin, M. A. Bauer, N. S. McIntryre
The University of Western Ontario, WSC-143, UWO, London, ON. N6A 5B7, Canada.

Abstract

The analysis of synchrotron-based Polychromatic X-ray Microdiffraction (PXM) data has been used by scientists and engineers to understand elastic and plastic strains in materials on a micro or nano scale. Such experiments generate hundreds or thousands of images where the analysis of each image often entails intensive computations- a challenging task. As well, in the past, the speed of such computations has made it difficult to obtain feedback on the experimental results in near real time. This has constrained researchers from making critical decisions on direction subsequent experiments should take based on the results in hand. In order to improve the analysis performance of PXM images, we have investigated the use of parallel analysis schemes. This paper reports on the design and implementation of accelerated PXM analysis software. It has been developed on IBM PowerXCell 8i processors and Intel quad-core Xeon processors. A substantial improvement in processing speed has been obtained to the extent that it should be possible to obtain results at the same rate as they are produced on the VESPERS beamline at the Canadian Light Source (CLS) Synchrotron (~1 Hz).

INTRODUCTION

The development of high-energy PXM as a non-destructive method to determine elastic and plastic strains has been ongoing for the past decade [1-5]. The data generated in PXM experiments can consist of a large number of 2D digital images. Once these images have been generated from an experiment, ideally, it is expected that data can be processed at a same speed level as data is collected.

There are three major procedures involved in PXM data analysis, including peak searching, indexing and strain calculation. Briefly, *peak searching* attempts to extract useful information about intensity points (peaks) from an image to be used as input for the next two procedures. The *indexing* procedure takes the output from the *peak searching* procedure and generates the structural information about the sample material, e.g. the orientation of a crystalline lattice plane from which a diffraction spot is generated. Based on the indexing results and peak information, the strain analysis procedure then produces strain tensors in the sample. Based on the indexing results and strain tensor information, an orientation map and a strain map can be generated for the entire scanned area from which all PXM data were collected.

There are some existing software packages for PXM data analysis, such as the 3D X-ray Micro-diffraction Analysis Software Package at APS in Chicago which was developed at ORNL[6], and X-ray Micro-diffraction Analysis Software (XMAS) at ALS in Berkeley[7]. The common feature of these two packages is that they both are Windows-based software with a frontend interface implemented in Interactive Data Language (IDL) [8] and some backend procedures implemented in Fortran. Both can process a large amount of PXM data sequentially, i.e., step by step and one by one in sequence. This is a very time consuming process, and it usually takes days to finish processing a set of data collected from one PXM experiment. However, synchrotron time is valuable and it is often difficult to get a scheduled beam time. Data analysis using existing software means that researchers must complete the analysis following their time on the synchrotron. Faster analysis could help researchers make decisions on subsequent experiments during their synchrotron session and gain significant insight into the materials that they are studying.

In this paper, we introduce the development of an accelerated software for PXM data analysis, so called Fast Online X-ray Micro-diffraction Analysis Software (FOXMAS). It has been developed on a Cell accelerator platform comprised of Intel and IBM Cell processors. The software developed and the system it runs on makes it possible for PXM data to be processed in "near-real time", that is, nearly as fast as it is being produced. A description of the platform, the development approach, some performance evaluations, conclusions and future work are reported.

CELL ACCELERATOR PLATFORM

The target Cell accelerator platform, called *Prickly*, is one of the clusters in SHARCNET [9]. It is a heterogeneous High Performance Computing (HPC) system consisting of one head node for hosting user logins and a chassis with 12 Linux cluster blades providing total 160 computing cores. Among the 12 blades, four blades are Intel blades and the other eight are IBM Cell blades. On each of the Intel blades, there are *two* quad-core Xeon E5420 processors running at 2.5GHz with 8GB of memory. Each of the Cell blades contains *two* PowerXCell 8i processors, so called Cell processors, running at 3.2GHz with 16GB of memory. Blade interconnection is achieved through Gigabit Ethernet.

Unlike traditional multi-core processors which are homogenous, such as those on Intel blades, the Cell processor itself has heterogonous multi-cores [10]. It employs two types of cores optimized for different kind of tasks. Each Cell processor has nine cores, i.e. *one*

*Work is part of Science Studio project supported by CANARIE http://www.canarie.ca/

PowerPC Processor Element (PPE) and *eight* Synergistic Processor Elements (SPEs). The PPE is just a traditional 64-bit Power processor and acts as a large-scale processor core to run the operating system and performs control-intensive tasks. In contrast, the SPEs are much simpler, but devote more resources to perform computationally intensive tasks. Since each Cell blade has two Cell processors, in total, there are *sixteen* SPEs on each Cell blade. The sixteen SPEs are independent, 128-bit vector processors. Each SPE has its own local storage (256KB) for instructions and data. The SPE access to the memory is achieved through its Direct Memory Access (DMA) controller. The DMA can work concurrently with SPE executions, which hides the latency caused by memory accesses.

The Element Interconnect Bus (EIB) provides four 128-bit data transmission channels for the inter-communication among PPE, SPEs, main memory and I/O. It can support up to 307GB/s bandwidth between any two bus units. Therefore, with EIB, each SPE can not only work alone, but also be chained together to perform data processing with an intensive workload, such as stream processing.

While the Cell's special architecture offers many advantages for high performance computations, the architecture also makes programming on Cell more difficult.

DEVELOPMENT APPROACH

The goal of this development is to port the PXM data analysis software onto the target Cell accelerator platform to achieve an accelerated performance.

There are two major challenges involved in this porting process. First, the exiting software was written in IDL with some backend procedures written in Fortran. Our target Cell platform Prickly can only support programs mainly in C/C++. The software has to be rewritten into C in order to make it run on Cell.

Another challenge is to program on the Cell. To make use of all those advanced features provided by the Cell, especially the computation power provided by those SPEs, programming on Cell is a challenging. As each SPE has its own local store for holding instructions and input/output data, data needs to be moved back and forth between the local store and the main memory with explicit DMA commands. Because of the limited space (256k) for a local store on SPEs, only tasks that fit can be considered, otherwise, an advanced overlay management needs to be used.

There were two objectives in developing the PXM data analysis application. First, we wanted to create an implementation of the three major analysis procedures where the processing tasks were pipelined in order to accelerate the processing of a PXM image. Second, we wanted an implementation so that multiple PXM images could also be processed in parallel.

To further improve the processing speed on a single image, we want to identify the performance "bottleneck" of the entire process and then target an implementation on Cell around that "critical" part. Our measurements on a sequential version of the analysis code indicated that more than 80% of the processing time was spent in the peak searching procedure; therefore, it was initially targeted as the "critical" computation to be considered for porting to the Cell.

The peak searching procedure involves finding a threshold, blob searching, and curve fittings on each of the blobs. Among all three subtasks in peak searching, curve fitting is the most intensive one. During curve fitting on a blob, it applies two 1-D fittings (i.e., one for the X direction and one for the Y direction) and one 2-D fitting for a box area around each blob. Fig. 1 illustrates blobs identified in a PXM image with a certain intensity threshold. Each fitting process actually entails solving a multi-variable, non-linear least square minimization problem. It involves iterations to update the state of corresponding variables continually until certain criteria

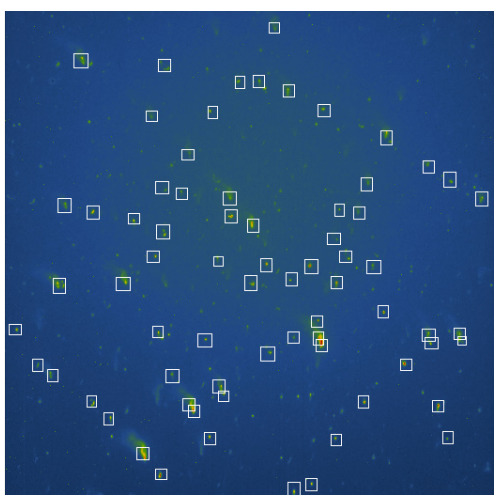


Figure 1: An PXM image with identified blobs that need curve fittings

are met. Specifically, the 1-D fitting involves solving four variables; these results become the initial states for the 2-D fitting. In turn, the 2-D fitting involves solving for six variables. The existing software carries out the curve fittings sequentially for each of the blobs in an image; this is very time consuming and becomes the bottleneck of the entire PXM data analysis.

Considering the computational power of a Cell's SPE, with a limited local store, it works well for a process with relatively small size but needs to run many times. Fortunately, the curve fitting is applied to each blob, which is in a relatively smaller area than the entire image area. The computation of the fitting process is also relatively intense and needs to be applied to every blob in an image. Therefore, the curve fitting process was selected as the processing task for the Cell's SPEs. After a collection of blobs has been identified, the fitting process can be done on the Cell's multiple SPEs in parallel, i.e. multiple blobs can be fitted by multiple SPEs simultaneously.

To analyze a large set of PXM images, we considered a computational approach involving processing multiple images in parallel and doing curve fitting on multiple blobs in parallel for each image during peak searching. The design of our parallel PXM data analysis program is illustrated in Fig. 2.

Using Fig. 2 as a guide the processing proceeds as follows. Initially, *n* images are loaded to be processed in parallel. Each of these images is initially processed on one of Cell's PPEs in order to identify blobs – potential regions having peaks. A list of blobs is produced for each image. Curve fitting is then done on each list of blobs on Cell's SPEs in parallel. The processing of the blobs from an image results in a list of possible peaks. The list of peaks is then passed to the indexing computation which results in index data and orientation maps based on index data. The index data is also used in the strain computation which produces a strain results and maps based on strain data. The indexing computation and strain computations are done on the Intel blades.

The design illustrated in Fig. 2 that has been implemented and deployed on Prickly, FOXMAS, has a web interface for job submission and online result visualization, that are not discussed in this paper. Because of the limited length, this paper only focused on the development of the accelerated data analysis.

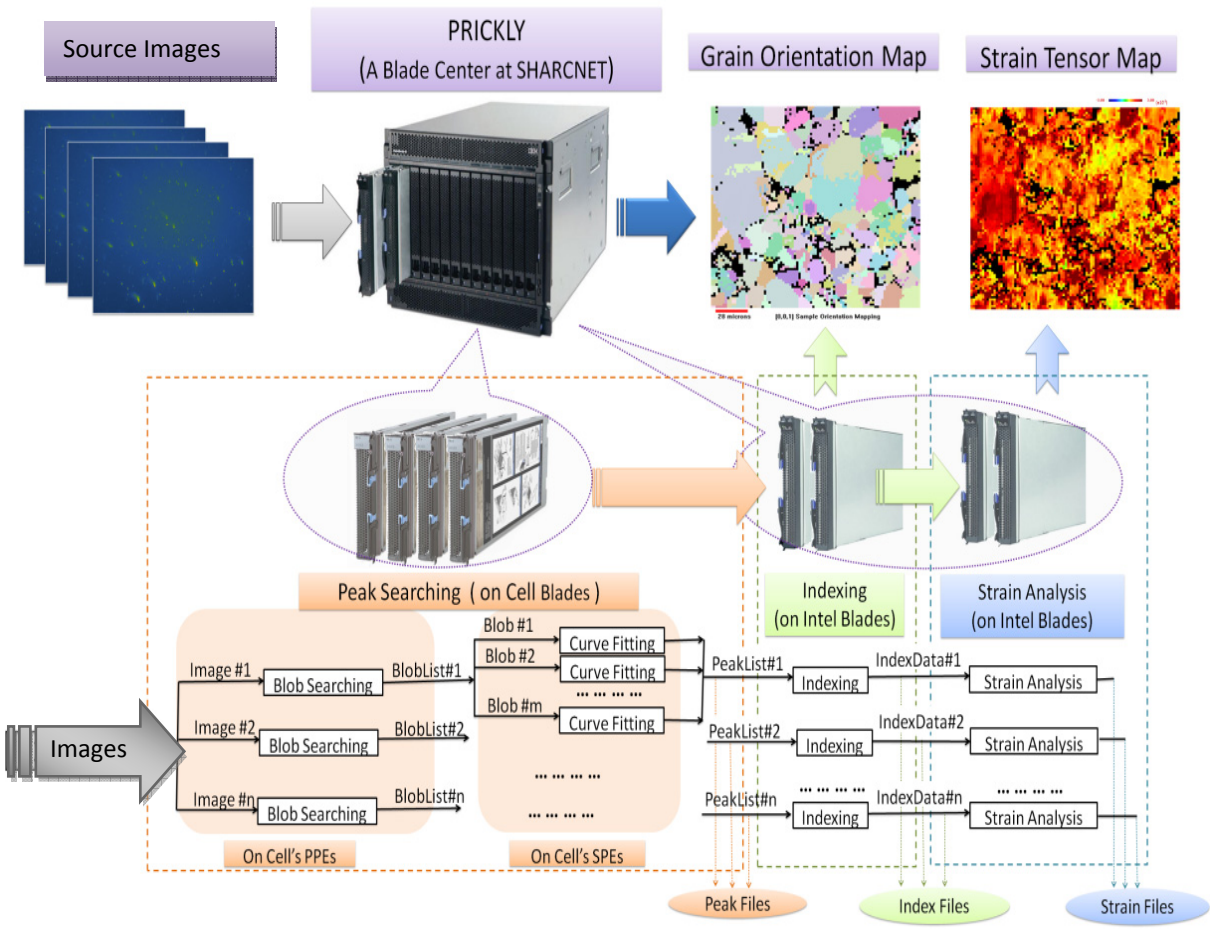


Figure 2: The configuration of high performance PXM data analysis on Prickly

PERFORMANCE EVALUATION

We measured the time needed for processing different sets of images sequentially with the original IDL software. We also measured the performance of FOXMAS, i.e., the average processing speed with various settings, including the number of parallel pipelines and number of SPEs used on each image. The speedup of each test case is compared to the speed of the IDL software.

PXM images could have different sizes depending on the type and setting of a CCD detector. Larger size images tend to provide more information with a trade-off of a heavier analysis workload. In this evaluation, we examined two different image sizes. One set of images were collected from APS, each of which has 1042X1042 pixels and is about 2MB/image. Another set of images were from CLS, each of which has 2084X2084 pixels and is about 8MB/image. Using the original IDL software on a desktop machine, the average processing speed for APS images is about *4.31* sec./image and for CLS images is about *14.36* sec./image.

Prickly has total of 4 Intel blades and 8 Cell blades, we examined the performance of data analysis on *one* pair of Cell-Intel blades and on *multiple* pairs of Cell-Intel blades. As described in Fig. 2, peak searching is done on the Cell blade; while indexing and strain analysis are done on the Intel blade. Since each Cell blade has a total of 16 SPEs, if n images are processed in parallel, i.e. n pipelines, and m SPEs are allocated for each image or each pipeline, m and n are constrained to be values such that $m \times n = 16$. Different combinations of m and n were tested. For the workload on the Intel blade, if n pipelines are initiated for processing n images in parallel, n processes are created on the Intel blade, and each of these n processes works on the indexing and strain analysis on one of n images. The operating system takes care of workload distribution among the *eight* cores on the Intel blade. The speedup of each test case is compared to the desktop speed using IDL software. Tables 1 and 2 present the measured results on one pair of computation nodes on Prickly.

Table 1: Results of processing APS images on *one* pair of Cell-Intel nodes on Prickly

Images in parallel (pipelines)	Number of SPEs for each image	Average Speed (sec./image)	Speedup vs. IDL (times)
1	16	0.63	6.84
2	8	0.43	10.02
4	4	0.35	12.31
8	2	0.26	16.58
16	1	0.22	19.59

The results presented in Table 1 illustrate that for processing images of the size of the APS images, the more images that are processed in parallel, the better throughput, i.e. processing 16 images resulted in average speed 0.22 sec./ image and speedup of 19.59 times compared to IDL software.

Table 2: Results of processing CLS images on *one* pair of Cell-Intel nodes on Prickly

Images in parallel (pipelines)	Number of SPEs for each image	Average speed (sec./image)	Speedup vs. IDL (times)
1	16	2.84	5.06
2	8	1.81	7.93
4	4	1.55	9.26
8	2	1.67	8.60
16	1	1.68	8.55

In processing the larger size images, i.e. those from CLS, the results of Table 2 suggest that processing 4 images in parallel and with 4 SPEs allocated for each image can produce the best throughput, i.e. an average speed 1.55 sec./image and about 9.26 times of speedup compared to IDL software. The processing of multiple images in parallel was achieved through multi-process programming, while the parallel blob fitting on the Cell was achieved through multi-threaded programming. In general, process creation cost is much larger than thread creation cost. The overall performance gain of a parallel application is dependent on balancing the computational workload and the trade-off in setting up the parallel processing elements. The result of reducing the number of SPEs allocated for each image, i.e., to 2 or 1 in Table 2, in lieu of having more parallel pipelines to process more images in parallel is not sufficient to overcome the blob processing done on each of the larger images within the SPEs. Consequently, using 4 pipelines and 4 SPEs results in the best performance for images of this size.

Table 3: Results of processing APS images on *multiple* pairs of Cell-Intel nodes on Prickly

Pair(s) of Cell-Intel nodes	Images in parallel (pipelines)	Average speed (sec./image)	Speedup vs. IDL (times)
1	16	0.22	19.59
2	32	0.14	30.78
3	48	0.07	61.57
4	64	0.07	61.57

To examine the performance of using *multiple* pairs of Cell-Intel nodes on Prickly, based on the results presented in Tables 1 and 2, *1* SPE per APS image and *4* SPEs per CLS image were used. Tables 3 and 4 present the results. The results illustrate that the performance of PXM data analysis has been boosted significantly when more computational resources are used. For the smaller sized images collected at APS, as presented in Table 3, when 48 processing pipelines were setup on *three* pairs of Cell-Intel blades, the average processing speed can reach as high as 0.07 sec./image, which is 61.57 times speedup compared with 4.31sec./image of using existing IDL software on a desktop machine. For larger size images collected at CLS, when 16 processing pipelines were setup on *four* pairs of Cell-Intel blades, the average

processing speed (see Table 4) can reach as high as 0.59 sec./image, which is 24.34 times speedup compared with 14.36 seconds/image when using the IDL software on a desktop machine.

Table 4: Results of processing CLS images on *multiple* pairs of Cell-Intel nodes on Prickly

Pair(s) of Cell-Intel nodes	Images in parallel (pipelines)	Average speed (sec./image)	Speedup vs. IDL (times)
1	4	1.55	9.26
2	8	0.96	14.96
3	12	0.70	20.51
4	16	0.59	24.34

Notably, the measured *speedup* from both sets of experiments presented in Tables 3 and 4 do not result in a linear improvement as more nodes are added. One of the factors affecting the speedup is the increased overhead of in setting up more pipelines exchanging information across the two types of nodes on Prickly. Another factor that affects achieving a linear speedup is the data transfer and communication cost of the Gigabit Ethernet; as more processes are added there is an increase in communication.

The goal of this project is to make use of such an accelerated data analysis for a synchrotron beamline to achieve a real time experiment and data analysis. The Cell platform Prickly is located at The University of Western Ontario in London Ontario. A synchrotron beamline, such as CLS is located in Saskatoon Saskatchewan. To achieve a real time PXM experiment and data analysis, data collected at CLS needs to be transferred to UWO in an ultra high speed. CANARIE's cross country lightpath network can provide such an ultra high speed data transmission. By using CANARIE's dedicated lightpath we are able to complete such a scenario.

A preliminary functional test has been measured for such a scenario. It included a procedure of sending a set of total 100 PXM images (about 8MB/image) from CLS to UWO, then getting processed on SHARCNET's Prickly at UWO, and presenting final results at an FTP site for users to download. It only took around 4 min. to complete the entire procedure. In specific, it took about 2 min. for data transmission from CLS server to UWO server through the lightpath, and less than 1 min. for data transmission from UWO server to SHARCNET's Prickly through UWO's intro-network. It only took about 1 min. to finish the data analysis on Prickly and send the analysis results back to UWO server for users to download. Even thought there are still rooms for refinement, the performance is quite promising for a real time experiment.

CONCLUSION AND FUTURE WORK

In this paper we reported the development of an accelerated PXM data analysis, FOXMAS, on a Cell accelerator platform, i.e. the cluster Prickly on SHARCNET. Using the computation power of Prickly, especially the Cell processors, FOXMAS can achieve up to 60 times faster than a desktop performance of using original IDL software package, depending on the size of images and the number of computation nodes used on Prickly. Combined with CANARIE's dedicated lightpath for data transmission, the promising performance makes it possible to process the data at the same high rate as it is produced at the synchrotron (CLS).

Future work for our next step is to implement the function of data transmission/analysis at the same time as it has been collected *during* a synchrotron experiment, i.e. a real time data collection and analysis. This is currently underway using the VESPERS beamline at the CLS. Such a model could also be adapted to other synchrotron and HPC facilities.

ACKNOWLEDGEMENTS

FOXMAS was developed based on source code of IDL packages from APS [6] and XMAS from ALS [7]. We are grateful for the useful document from Dr. Nobumichi Tamura of ALS in helping us to understand the analysis procedures involved in PXM data analysis. Thanks to Dr. M.L. Suominen Fuller and Ph.D. student Jing Chao at UWO for their great help on the validation of results produced by FOXMAS. Thanks to Dong Liu at CLS for his collaborated work in the measurement of data transportation from CLS to UWO through the dedicated lightpath.

REFERENCES

[1] J.S. Chung and G. E. Ice, *Journal of Applied Physics*, 86 (1999) 5249-5255

[2] G. E. Ice and B. C. Larson, *Advanced Engineering Materials,* 2(2000) 643-646

[3] B. C. Larson, W. Yang, G. E. Ice, J. D. Budai and J. Z. Tischler, *Nature,* 415(2002) 887-890.

[4] M. L. Suominen Fuller, R. J. Klassen, N. S. McIntyre, A. R. Gerson, S. Ramamurthy, P. J. King and W. Liu, *Journal of Nuclear Materials*,374 (2008) 482-487

[5] J. Chao, A. Mark, M.L. Suominen Fuller, N. S. McIntyre, R. A. Holt, R.J. Klassen and W. Liu, *Material Science and Engineering,* A 524 (2009) 20-27

[6] PXM data analysis at APS : http://www.aps.anl.gov/Sectors/33_34/microdiff/downloads/

[7] PXM data analysis at ALS: http://xraysweb.lbl.gov/microdif/user_resources.htm

[8] K. P. Bowman, *An Introduciton to Programming with IDL: Interactive Data Language*, Elsevier Inc. 2006

[9] SHARCNET website: https://www.sharcnet.ca/

[10] M. Scarpino, *Programming the Cell Processor: For Games, Graphics, and Computation,* Printice Hall, 2008

FAST ORBIT CORRECTION AT THE CANADIAN LIGHT SOURCE

C. Payne, Canadian Light Source, Saskatoon, Saskatchewan, Canada
D. Chabot, Brookhaven National Laboratory, Upton, New York, USA

Abstract

Correction of the electron beam orbit in the storage ring at the Canadian Light Source has historically been implemented using a correction system capable of only moderate update rates. Over the past several years work has been undertaken to reduce orbit perturbations and improve end user synchrotron beam quality by reimplementing the correction system and enabling orbit corrections several orders of magnitude faster. This paper will describe the implementation and migration of the orbit control software from the slow correction system to the fast system.

CLS ORBIT CONTROL HISTORY

The present orbit control system in use at the Canadian Light Source (CLS) is described in [1]. This system is an intermediate step between the previous orbit control system [2] and the system described in this paper. The design limitations of the current system as impetus for change are worth mentioning and will be briefly discussed below.

DESIGN LIMITATIONS OF THE EXISTING SYSTEM

There are several key limitations inherent to the orbit control system in use at the CLS at the time of writing.

Update Rate Limitations

The main motivational factor to migrate to a new system is maximum possible update rate. The present system is only capable of quasi-static update rates on the order of 0.1Hz. Although this has been successful at sufficiently maintaining the orbit of the CLS Storage Ring (SR), faster corrections rates are desired to further reduce orbit perturbations.

Serial Application of Orbit Corrections

The Matlab [3] program, CLSORB [4], applies corrections in a sequential manner. This results in undesirable orbit perturbations as the corrections are applied one after another around the storage ring. Distribution of corrections from CLSORB through the Experiential Physics and Industrial Control System (EPICS) [5] produces additional non-deterministic behaviour due to network and computer latencies. Delivering corrector magnet setpoints in this way also has the effect of accruing hardware delays on a per-channel basis, instead of per power-supply controller. This adds significant delays to the process of setpoint distribution, and is a major factor governing the achievable rate of the orbit control system.

HARDWARE

The hardware involved in the orbit control system is shown schematically in Figure 1. The hardware consists of:

- An Industrial 3GHz x86 PC IOC with 1GB or RAM running Real-Time Executive for Multiprocessor Systems (RTEMS) v4.10 [6]
- Four (4) Versa Module Eurocard (VME) Crates [7]
- Four (4) pairs of Struck Innovative Systems (SIS) PCI/VME 1100/3100 cards, for connectivity [8]
- Four (4) Analog to Digital Converter (ADC) VME cards (ICS-110BL sampling ADC)
- Eight (8) Digital I/O Modules, model VMIC 2536 D-I/O, 2 per VME crate used to control corrector setpoints

In addition, hardware independent of the fast orbit control software system:

- Beam Position Monitors (BPMs) which produce analog signals in proportion to the position of the electron beam passing through them. [9]
- Bergoz BPM Modules which sample the BPMs to produce analog x-y coordinates of beam position. [10] These signals are then digitized by ICS-110BL VME modules and the data processed by the RTEMS IOC.
- OCM Power Supply Controllers, VME based devices interfaced with the OCM power supplies. There are 48 vertical and 48 horizontal orbit correctors, contained in a bulk IEPower [11] chassis. It should be noted that although setpoints are via the fast, VME interface, the power supply feedback is exclusively via serial interface.

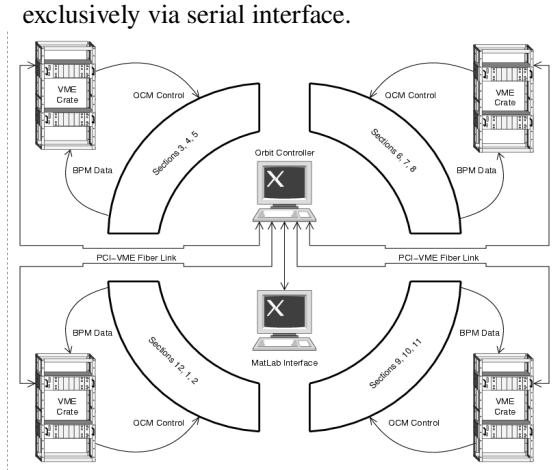


Figure 1: Hardware Overview

ADVANTAGES OF THE FAST IMPLEMENTATION

Although the current orbit control system has served the CLS well, a more efficient and faster system has been designed. Along with speed, there are several other advantages of the fast orbit control system over the existing system.

Update Rate

As the name implies, the key advantage of the new system is an improved correction rate, which will be several orders of magnitude faster than the existing system. With the fast system, rates of 20-100 Hz are attainable.

Concurrently Acquired BPM Data

BPM data acquisition in the fast system can be period driven rather than interrupt driven. Although this shift from interrupt driven is necessitated by the inability of determining detailed information on the FIFO buffer state of the ICS-110BL, it has several, positive side effects. Using per ADC threads (4) to concurrently acquire the BPM data ensures true time correlation amongst the BPM data. As well, the threaded, concurrent nature of the BPM data acquisition results in both a reduction in noise by approximately a factor of 4 as well as a reduction in dead time of a factor of 16. [1]

Localized Setpoint Calculation and Application

Previous versions of orbit control software have relied on multiple computers working together to create a complete orbit control system. One system would acquire the orbit positions, pass them to another system, which would calculate and apply the corrections, either back to the initial system, or to other slow systems.

Corrections are now calculated and applied directly by the RTEMS Input Output Controller (IOC). This migration from a remote setpoint application, with network and other latencies involved, greatly increases the rate at which applications may be applied to the system.

In the fast system, once the response matrix is determined and transferred from Matlab to the RTEMS IOC, the system operates independently, calculating and distributing new corrector magnet setpoints.

Concurrent Correction Setpoint Application

Similar to the concurrent acquisition of BPM data, the fast system is also able to simultaneously apply the correction setpoints to the orbit correctors. Once the setpoint values are calculated and loaded to the corrector controllers, they are simultaneously activated across all controller channels. This method minimizes the perturbations caused by the serial application inherent in the previous system.

Multiple Operating Modes

The fast system has the added flexibility of allowing multiple modes of operation. [12] This feature is useful both for initial commissioning as well as providing a migration path from the current system to the fast system. The modes, which are exposed and controlled via an EPICS Process Variable (PV) include:

- **Standby**: In this mode, BPM data acquisition is disabled, but control of corrector setpoint PV's is permitted.
- **Assisted**: This mode causes the system to emulate the current system where BPM process values are averaged and updated at 20Hz and corrector setpoints are distributed via the CLSORB Matlab program. Assisted mode provides a migration path between the current system and the fast system, and has been in use since April 2010 while awaiting delivery of a full compliment of fast VME controller boards.
- **Autonomous**: In this mode BPM acquisition is identical to Assisted but the RTEMS IOC will calculate and apply corrections independent of CLSORB.
- **Timed**: This mode is similar to Autonomous, but the BPM data acquisition is timer driven thereby allowing faster operating rates.

Of these modes, Autonomous and Timed will be the modes used most often during normal operation.

Orbit Control EPICS Interface

The fast system also exposes various orbit control parameters as EPICS PV's, leveraging the nature of the Channel Access (CA) protocol to provide a user interface accessible to distributed CA client programs. The accessible system parameters include BPM x and y averages and standard deviations, the number of BPM samples per average, as well as corrector magnet setpoint control and read-backs.

SYSTEM PERFORMANCE

Initial testing performed in March 2009 with one half (48) the total required number of fast correctors available has shown promising results. The test consisted of opening and closing the gap of the Hard X-ray Micro-Analysis (HXMA) Wiggler while observing the expected orbit perturbation. Historically this is a disruptive operation, causing relatively large orbit excursions. The effectiveness of the new orbit control system's ability to dampen beam disturbances is readily apparent from Figure 2 and Figure 3 below.

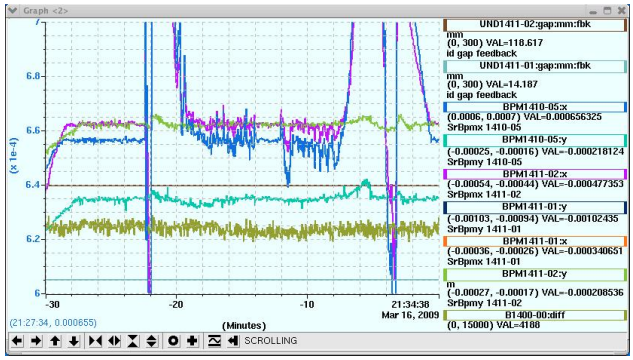


Figure 2: Orbit response due to HXMA Wiggler Operation while operating orbit control in Assisted Mode, equivalent to slow operation.

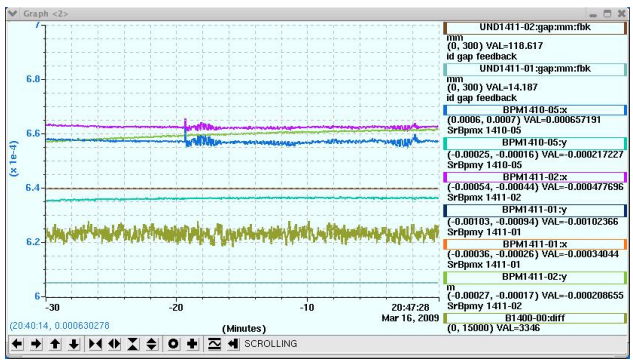


Figure 3: Orbit response due to HXMA Wiggler Operation while operating orbit control in Timed Mode, Approximately 65Hz update rate.

Theoretical limits to the update rate of the fast system in the current configuration are on the order of 100Hz. Realistically attainable rates in Timed mode are 65Hz.

FUTURE WORK

Preliminary work has been done on the next generation of fast orbit control which will be capable of even higher update rates. The changes required to attain even faster rates include heavily modifying the power supply setpoint distribution algorithm to distribute the cost of the hardware delays associated with affecting setpoint changes. As well, a behavioural modification permitting application of the orbit control algorithm based on the number of ADC frames collected, rather than based on ADC or RTEMS timer interrupts has been implemented. Coupled together these changes permit application of orbit corrections well in excess of 100 Hz.

CONCLUSION

The CLS storage ring orbit control has been driven by CLSORB for several years. Although sufficient in controlling the orbit to allow storing beam for long periods of time, the system does not operate fast enough to counteract insertion device movement or other sources of beam disturbance on the order of 10 Hz.

The benefits of an improved orbit correction system for the CLS storage ring are obvious from the example given. Routine activities such as movement of insertion device gap can be made almost transparent to user operations. Other, low-frequency sources of beam disturbance can also be effectively compensated for with the new system.

Fast correction hardware availability has been the limiting factor in full deployment of the fast orbit correction system. We are hopeful that the system will be fully deployed in late 2010.

ACKNOWLEDGEMENTS

Research described in this paper was performed at the Canadian Light Source, which is supported by the Natural Sciences and Engineering Research Council of Canada, the National Research Council Canada, the Canadian Institutes of Health Research, the Province of Saskatchewan, Western Economic Diversification Canada, and the University of Saskatchewan.

REFERENCES

[1] D. Chabot, “SR1 Orbit Control Design Note”, 5.2.39.4 - Rev. 0, 2008
[2] R. Berg, “Orbit Control for the Canadian Light Source”, EPAC Proceedings 2004
[3] Matlab http://www.mathworks.com/products/matlab/
[4] H, Zhang “CLSORB Slow SR Orbit Control in Matlab”, 7.2.61.6 Rev. 0 , Jun 2008
[5] Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/
[6] Real-Time Executive for Multiprocessor Systems, http://www.rtems.com
[7] Versa Module Eurocard, http://www.vita.com
[8] Struck Innovative Systems, http://www.struck.de
[9] D. Bertwistle, CLS Design Note 7.2.38.1 Rev. 0, “CLS Button Position Monitor Sensitivity Analysis”, Nov 2001
[10] I. Stavness, CLS Technical Document 7.2.38.5 Rev 0, “BPM Test Report”, Sept 2003
[11] IEPower Inc, http://www.iepower.com
[12] D. Chabot, “CLS Orbit Control Notes, Sept 2009, unpublished

PRESENTATION ONLY

WHAT'S BEHIND AN ACCELERATOR-CONTROL SYSTEM?

Rüdiger Schmitz, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany.

Abstract

A control system has a lot of features, some essential: e.g. a set of application programs. The infrastructure they need in order to run so that the operators at least be able to switch the accelerator on and off.

Graphical User Interfaces, intelligent control algorithms or data acquisition methods are obvious, but other features (not as obvious) also require considerable manpower and should not be underestimated. They have a major impact on the availability of the control system. I call these features the 'meta-control system'.

This paper describes the efforts made by the control systems group at DESY to provide a reliable tool for the operators, minimizing the downtime caused by control system failures. It reviews this aspect of computer based accelerator control dating back to the late 1970s when the accelerator PETRA went into operation, controlled entirely by mini-computers from Norsk Data [1].

Both the computer with the supporting technology and the control system group are essential to an accelerator's success.

INTRODUCTION

MCS -the machine control group at DESY- has built, maintains and improves the control systems of all current DESY-accelerators: The preaccelerators LINAC II, DESY II and PIA, the light sources DORIS III and PETRA III and the free electron laser light source FLASH II. Since the decision to switch off the proton-lepton collider HERA II in 2007, DESY changed its scientific profile from a predominantly high-energy physics laboratory to a synchrotron light research centre. This had a major impact on the required reliability and availability of the control systems:

- The top-up mode for PETRA III does not tolerate any failure in the accelerator-chain for more than 5 minutes.
- The cramped schedule of the beam line experiments at DORIS III, PETRA III and FLASH II may well leave behind an unhappy user if part or all of the requested beamtime is lost.

OPERATOR VIEW

A control system is most visible at the operator-console. Nowadays this is an assembly of monitors and input devices such as mice, knobs or keyboards connected to computers. The operator console is the place from which all available functions of the accelerator in its different phases of operation can be controlled: user run, maintenance periods and machine studies.

The technical implementation differs a lot from control system to control system, but nevertheless the look and feel is not much different. (FLASH and PETRA have different 'control systems' but for some areas like vacuum and sequencer there is hardly any difference.) Differences arise from the different age of the accelerators and also from the skills and preferences of the constructor and operator.

Application programs in operator consoles may be rich clients written in Java and Visual-Basic or they may be operator panels generated for example by jDDD and web-based-applications running in a browser such as Web2C [2].

At the other end, there is the accelerator which will be directly affected by actions initiated at the operator-console or by automatic processes running independently of operator interaction. The diagnostic- and machine-protection systems will necessarily report any malfunction of control system procedures.

In between we have what I call here a 'communication cloud', i.e. something allowing communication between the operator console and the accelerator. This leads to the simple operator view of an accelerator control system shown in Fig. 1.

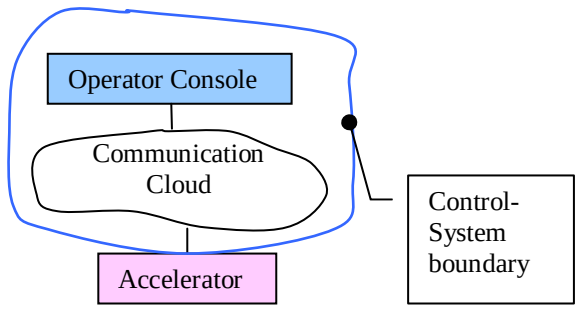


Figure 1: Simple operator view of a control system

CONTROL SYSTEM PEOPLE VIEW

Looking more deeply into a control system one can identify the different hardware building blocks in the 'communication cloud': Computers, networks, field buses, diagnostic systems and turnkey systems.

There is no precise definition as to where control systems boundaries are drawn. What belongs and does not belong to the control system is defined in different ways by different people. But at least one needs all major subsystems interfaced to the control system.

An even deeper view will bring us to the software. But at this point the system cannot be understood without yet further information, information which cannot necessarily be found in the control system.

To get information about the underlying principles and concepts of a control system one should ask the control system group. They will use a lot of buzzwords or

abbreviations and will refer to documentation web-sites or talks given at conferences. They will use the term ‘control system’ in two different ways:

(1) A system controlling an accelerator, e.g. ‘The control system for PETRA III’
(2) A name for a set of tools providing communication protocols and services which make for efficient client and server applications; e.g. TINE, DOOCS, EPICS, TANGO [3]

I prefer the use of the term ‘control system’ as defined in (1). Running a control system involves more features than are laid down in the documents mentioned above. Fig. 2 illustrates this fact:

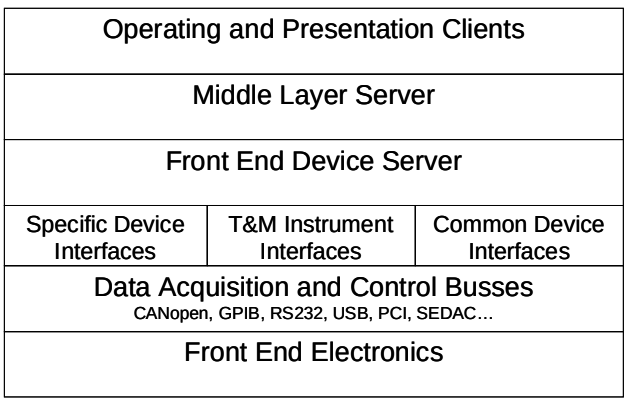


Figure 2: Architecture of the control system for PETRA III [4]

Why is Fig. 2 incomplete? A few examples:

- The daily operational needs may require minor or major improvements, perhaps even adding to or redesigning a certain feature. This may lead to pragmatic solutions not foreseen in the description of the tool-sets.
- There are old systems which, due to financial and/or manpower bottlenecks cannot be upgraded.
- The control system has to be able to cope with failures and unforeseen situations.
- The technical and personal environment of the control system changes.
- The diagram shows the system under normal operation conditions, but not the process of achieving these conditions (e.g. when the system is installed) nor the maintenance operations required during the lifetime of the accelerator (e.g. when hard- or software is replaced).
- Some purely pragmatic modifications might lead to a situation in which the diagram no longer represents reality.

Over and above the technical changes just outlined, the control system people are faced with a number of jobs to do and problems to solve. I describe these jobs as the ‘meta-control system’.

META-CONTROL SYSTEM VIEW

Here I describe some of the features that a meta-control system should have. The focus is on the control systems of PETRA III and DORIS III and its preaccelerators, using TINE as an integrating tool-kit.

Fault detection and repair

The following features or improvements have been added or made since 2005 [5]:

- JAVA-Applications write Logging-Information. The control group at DESY has a Log-Viewer-Application to identify faulty applications. Until now, however, there is no automatic notification. (jDDD has recently set a notification to a Java Message Service Server.)
- Remote control of Device-Servers shows status and allows restart.
- What we have named ‘Spider’ shows the status of all links to TINE network devices. A ‘Tarantula’ crawls through those links from one level down to the next, building a tree of dependencies.
- For each type of Windows host used in the control system a spare is kept running and there is an agreed procedure for how to replace a broken computer with the appropriate spare. The time needed for the replacement is about 30 minutes.

Control systems central database

Measures are in place to make our system resilient to disk crashes or computer breakdowns

The configurations and initialisations of Device Server Computers as well as the processes they host are laid down in a Central File Repository. There are semi-automatic procedures using this information to setup a new or replace a broken computer. Work still needs to be done in order to ensure support for different operating systems with the same Central File Repository (e.g. proper choice of file-transfer modes, OS-independent file-formats).

At regular intervals a central upload process copies local files to a network repository. From there they will be downloaded during the setup process mentioned above.

Application deployment

We have implemented a ‘build and deploy’ procedure for JAVA applications [6] which eases the work of the programmers and enforces our guidelines for the creation and storage of JNLP files. (An offline-tool for re-checking the files has yet to be integrated into ‘build and deploy’.)

Application programming policies

A rich client application written for example in JAVA can make use of all features of the language; but we have rules which restrict usage of the features and it is the programmers’ responsibility to obey them. So the control-people may never lose sight of – or allow others to lose sight of – these policies.

An application-framework [7] helps to create applications which are policy conform, and operator panels generated by a panel editor such as jDDD can fully implement the policies.

Maintenance strategies

The control group uses the same operating system updating procedures as the DESY central IT, but provisions have been made for the control group to trigger the process only when it fits into the accelerator schedule.

Proactive maintenance is achieved by looking for critical events in system-log files (DISK-errors, unscheduled reboots). Preventive maintenance involves replacing equipment before end-of-life.

Defence against attacks

DESY's central IT-Group provides and maintains the anti virus software used by the Windows PCs of the control system networks. These networks have no or only limited access to the internet. Access-lists in the control system network routers which could help to protect the control system are under construction.

Access to accelerator equipment is secured by a device-server specific list of ip-addresses and accounts.

Monitoring the system

We recently implemented monitoring tools servers which are fully integrated into the tool-set TINE. e.g.: a **locator service** shows the location of all network-devices connected to the control system-network. This will automatically trace roaming equipment such as vacuum pumping stations. A **network analysis service** gives information on bottlenecks or bad network connections and may generate an alarm.

Integration into Campus IT-Infrastructure

We try to use as many central services as possible. Some services have been introduced by us in close cooperation with the central services people. We have to keep an eye on how the control system is affected when any of the above is out of order.

This strategy has been positive both for us and for the machine-physicists and service-people who easily exchange data between the control system and their workplace.

Observation of hard- and software life cycles in relation to the accelerator's lifetime

A regular review of control systems is advisable, keeping an eye on software- and hardware-lifetimes and on possible improvements or necessary renewals.

The end of support for hard- and software forces us to make decisions. For example Microsoft Windows XP extended support ends in 2014, thus we will not and do not need to upgrade the control system of DORIS III to Windows 7 because the operation ceases by end of 2012.

Choice of adequate hard- and software-solutions

We mainly use standard PCs in the control system, and only occasionally avail ourselves of more expensive solutions. We have generally had success with this policy. It saves money and keeps diversity low.

The chosen hard- and software solution must meet the requirements and the skills and experience of the control system people.

Preserving approved concepts

In 1978 we operated PETRA I with mini-computers. We had implemented many of these meta-control system features. They disappeared along with the computers and, at least some, out of the heads of the colleagues! I think good concepts and their principles should be preserved.

CONCLUSION

An accelerator-control system should support the reliable operation of an accelerator in all its different operational phases with as few interruptions as possible.

The control systems group is responsible for that job, formulating and activating the concepts, policies etc. which hold the control system together and defending it against various quick fix pseudo solutions, which are so often proposed. Indeed, the control system people are the custodians of the meta-control system!

I believe that, in any institution, you will have at least as many control systems as there are control-groups, even if there are no or only slight technical differences. The control systems for DORIS III and PETRA III for instance are technically quite different but are maintained by the same people and the meta-control system is therefore the same.

So what is behind an accelerator-control system? The control system group!

REFERENCES

[1] Norsk Data, Oslo, Norway, ceased to exist 1992
[2] http://jddd.desy.de http://web2ctoolkit.desy.de
[3] http://tine.desy.de http://www.aps.anl.gov/epics http://doocs.desy.de http://tango-controls.org
[4] From: R.Bacher, "The New Control System for the Future Low-Emittance Light Source Petra 3 at Desy", Proceedings of EPAC 2006, Edinburgh.
[5] P.Duval U.Lauströer R.Schmitz, "Fault Identification in Accelerator Control", Proceedings of PCaPAC 2005, Hayama, Japan.
[6] A.Labudda, "Building and Deploying loosely coupled Console Applications", Proceedings of PCaPAC 2006, Newport News, USA, p 126
[7] K.Hinsch and W.Schütte, "MstApp, a Control Application Framework at DESY", to be published in 2011. See also: J.Wilgen, "First Experiences with a Device Server Generator for Server Applications for Petra III", Proceedings of PCaPAC 2008, Ljubljana.

TANGO COLLABORATION NEWS

J. Meyer (ESRF) on behalf of the Tango community
ALBA, DESY, ELETTRA, ESRF, SOLEIL

Abstract

During the last years, the Tango collaboration was and is still growing. More and more users are requesting new features and developing new tools for Tango. Decisions whether the requested features will be implemented and whether new tools will be part of the Tango distribution need to be made. The organizational aspects of the collaboration need to be clarified as well as the decision making process for new developments.

This paper will explain the collaboration, its organization and the decision making process as well as the latest facts and features around Tango.

Some ongoing developments are the new code generation tool to allow inheritance in the Tango class structure, the new event system for high bandwidth event distribution and the Tango packaging to allow installation with a few clicks.

WHAT IS TANGO?

Tango [1] is a control system tool kit developed by a community of institutes. It is object oriented with the notion of devices (objects) for each piece of hardware or software to be controlled. Tango classes are merged within operating system processes called Device Servers. Three types of communication between clients and servers are supported (synchronous, asynchronous and event driven).

But Tango is not only the software bus which handles the communication between device servers and clients. The Tango tool chain offers software from the hardware interface to the graphical user interface for several programming languages.

Tango utilities are available, with the basic installation, for code generation, device configuration and testing and for administration and survey of a whole Tango control system.

An archiving and a configuration snapshot system usable with Oracle or MySQL are also available.

Table 1 : Available Tango Modules

Module	Description
Core Libraries	Client/Server communication libraries for C++, Python and Java
Device Classes	About 300 hardware interface classes are available to download [1]
GUI Frameworks	Available for C++ and Python using QT, for Java using Swing and a web interface written in PHP
Client Bindings	LabView, Matlab and IgorPro
Tools	Pogo – Code generator for device classes in C++, Python and Java Jive – Configuration and testing tool Astor – Administration and survey of the Control system
Archiving	Archiving and snapshot system with GUIs and web interface. Usable with Oracle and MySQL
Alarm System	Event driven alarm service
Sardana	Framework for experiment control : Interface standardization, configuration, sequencing, command line interface

COLLABORATION HISTORY

Tango development started in 1999 at the ESRF. SOLEIL joined as the first partner in 2002, ELETTRA and ALBA joined in 2004 and the DESY (beamline controls) in 2008.

For every new member a new memorandum of understanding was signed by all collaboration partners.

We meet twice a year to discuss all ongoing projects. In case of lack of consensus, we tried to find a solution, all collaboration partners could agree upon.

A coordinator was named in each institute for all organisational, but also technical requests concerning Tango.

A mailing list is available for all questions and propositions to the whole Tango community.

A GROWING COMMUNITY

Since last year we have two new institutes requesting to join the collaboration: MAX-lab in Sweden, FRM-II in Germany. Tango is also used by other laboratories, for example LMJ (beam diagnostics) in France. Industrial companies are evaluating Tango, due to outsourcing requests from new projects.

The number of software development projects around Tango is increasing. To package the system and to keep the source repositories clean, we have to decide which projects will be part of the Tango distribution and which ones will be add-ons.

With the growing community, the increasing number of users and the foreseeable number of new developments around Tango, we have to find a new organisational form, to be sure, to take decisions on development priorities and strategies within a reasonable delay.

THE NEW ORGANIZATION

Taking into account the increasing number of users, we will reduce the frequency of Tango meetings to reduce organizational effort and cost. Specialised meetings on particular development projects are encouraged.

To allow a fast decision making process we are changing the collaboration management structure. The new structure has three levels:

1. The executive committee:

The executive committee takes the strategic decisions about developments in the Tango collaboration. There is one member from each institute who has signed the memorandum of understanding. This representative should have enough power to decide on allocating resources to develop software for Tango.

The committee will meet, at least, just after the Tango collaboration meetings.

2. The collaboration coordinator:

The collaboration coordinator is the central point of the organizational structure and liaises between the project leads and the executive committee. His responsibility is:

- To organize and coordinate the executive committee meetings, to produce a report of the committee meeting and to give feedback to the Tango community.
- To maintain a global project plan, in collaboration with the project leads, including requirements, schedule and resource requirements.

3. The project leaders

Besides the Tango core libraries, several packages are considered to be part of the Tango controls system. A list of these packages is maintained up-to-date by the collaboration coordinator and any change to this list is decided by the executive committee. Each package which is part of the Tango core has to have a project leader.

For the Tango community, the project leader is the contact person for all questions and remarks concerning that particular project. He is in charge of following the project schedule and ensuring the requirements are satisfied. In case of problems impacting on other Tango project(s), the project leader refers questions to the collaboration coordinator and eventually to the executive committee.

We distinguish two different collaboration membership types:

- **Committer**: must contribute resources to the collaboration. He is responsible for one or more Tango core packages.
- **Contributor**: can propose code modifications to the committers for the Tango core packages and submits Tango device classes to the public device classes repository.

How to accept a new collaboration partner?

To be an official member of the Tango collaboration, a new institute needs to sign the memorandum of understanding. New members are to be accepted with a unanimous decision by the executive committee.

How to get an official Tango decision?

All requests for decision should be sent to the collaboration coordinator. They will be presented to the executive committee during the next committee meeting.

Decisions are made by voting. The vote of each executive committee member is weighted according to its status as contributor or committer (cf. above). Each committee member has at least a weight of one. An extra vote is acquired if the committee member represents an institute which is also a committer.

ON-GOING PROJECTS

The Packaging

To allow an easy way to install and run Tango we prepare binary packages on top of the source code distribution.

A binary package is available for Windows, since a long time, from the Tango web site [1]. Now a first version of binary packages is available for Debian and Ubuntu Linux users. From Launchpad the different packages can be installed as needed [2]. Investigations are ongoing how to support binary packages for other Linux distributions.

The Tango Box

The Tango box is a virtual Linux computer which runs in the VMware Player [3] virtualisation software. On this virtual machine runs a Tango system and most of the Tango tools are installed and ready to be used. It offers an overview of the Tango software on a running system without installing Tango on a local machine. The software on the Tango box is updated once a year.

GUI Developments

A lot of effort is spent to add more features and new functionality to the available graphical toolkits.

The Python toolkit Tau supports the whole spectrum of viewers now. With the C++ toolkit QTango, synoptic displays can be created from CAD drawings. On the Java side, an on-going development will open the toolkit for different data sources. This should allow the usage of widgets with non Tango data sources.

Pogo the Code Generator

All the Tango classes follow the same skeleton. Therefore, a code generator (Pogo) has been written to generate these skeletons. This tool was available at the very beginning of Tango. Pogo was implemented using hand written parsing techniques. The decision was taken to re-write the code generator.

The new release of Pogo is based on modern techniques using Xtext [4] to create a Tango DSL (Domain Specific Language). This DSL is then used to describe the new Tango class. Using Xpand [5] and a set of templates, the Tango class skeleton is generated. Xtext and Xpand are part of the Eclipse modeling project [6].

With this new way of generating code, it is now possible to implement inheritance of Tango device classes properly. Only the inheritance of abstract interface classes was allowed before.

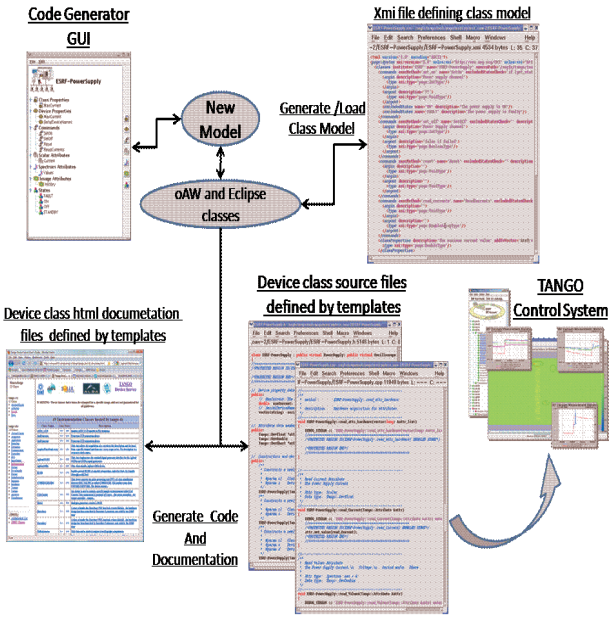


Figure 1 : Code generation with Pogo

The new code generator is available for generating C++ classes. The templates for Python and Java still need to be written.

THE NEAR FUTURE

A Faster Event System

The Tango event system is based on the CORBA [7] notification service, the implementation used is omniNotify [8]. Today's event rates are sufficient but cannot be improved due to the implementation of omniNotify (dead project). The detailed problems have been already described at ICALEPCS 2009 [9].

Performance measurements for event distribution have been carried out using the Data Distribution Service (DDS) [10] implementation OpenSplice [11] and the publisher/subscriber pattern of the ØMQ [12] Socket API.

The measured performance values are in received events per second between two machines (P4, 2.5GHz, Ubuntu 9.04 – Core 2 Duo, 2.6GHz, Ubuntu 9.04) on a 100 Mbit network.

Table 2 : Event System Performance Tests

Sub	1 int (32bits)			1024 int		
	Tango	**DDS**	**ØMQ**	**Tango**	**DDS**	**ØMQ**
1	770	12500	45000	650	1850	2400
5	400	7900	14000	200	1800	500
10	220	6500	7300	100	1700	230

DDS showed the best performance, for a growing number of subscribers, due to its multicasting protocol. But it has a set of drawbacks for programming and configuration. The ØMQ performance was measured only with unicast transmission because the multicasting showed reliability problems.

Table 3 : Event Systems Advantages and Drawbacks

	DDS	**ØMQ**
+	CORBA ORB/DDS cohabitation, Performance, QoS, Multicasting	No extra processes, Single cast performance, Can switch from uni- to multicast transmission
-	Three processes + shared memory per host, SIGKILL forbidden, No core dump, No dynamic data partitioning possible	Multicasting not yet 100% reliable, Young product, More integration code to write

The Tango philosophy is to keep it simple. ØMQ seems to be more adapted for us, even if the programming effort is higher and we have to collaborate with the implementers to make multicasting reliable. Due to the complexity of a multicasting set-up we would like to keep unicasting as the default transmission for the event system. But, multicasting should be available when needed.

Library for Image Acquisition (LIMA)

LIMA is a project for the unified control of two dimensional detectors. The aim is to clearly separate hardware specific code from the common software configuration and common features, like setting standard acquisition parameters (exposure time, external trigger, etc), file saving and image processing.

Requirements and specifications are actually collected from the interested institutes.

On top of the functionality of this library, a common Tango interface for 2D detectors should be defined.

REFERENCES

[1] http://www.tango-controls.org
[2] https://launchpad.net/~abogani/+archive/tango
[3] http://www.vmware.com/products/player
[4] http://www.eclipse.org/Xtext
[5] http://wiki.eclipse.org/Xpand
[6] http://www.eclipse.org/modeling
[7] http://www.corba.org
[8] http://omninotify.sourceforge.net
[9] E.Taurel, "Tango Kernel Status and Evolution", ICALEPCS'09, Kobe, Japan, 2009, THA001, p. 630 (2009); http://www.JACoW.org.
[10]http://www.omg.org/tlogy/documents/dds_spec_catalog.htm
[11] http://www.opensplice.org
[12] http://www.zeromq.org

THE TINE CONTROL SYSTEM PROTOCOL: HOW TO ACHIEVE HIGH SCALABILITY AND PERFORMANCE

P. Duval and S. Herb, DESY/Hamburg

Abstract

Over the years the TINE control system [1] has implemented numerous strategies for achieving high efficiency data transport within a distributed control system. This was essential for controlling a large machine such as HERA [2]. Our recent experience with controls for the PETRA3 and FLASH accelerator complexes at DESY has revealed new scalability issues. The principal problem has been in limiting the communications load on the front end servers and network in the presence of increasing numbers of client applications, many of which are written by 'part-time' developers who prefer simple API calls, or use development platforms which support only such calls. A single such application, polling hundreds of devices, may generate ~1000 calls per second to a single server. This load on the server can be reduced if, for example, the intermediate software layers can consolidate such calls into array transfers. TINE now offers various 'second-order' protocol features which go a long way toward not just allowing but 'enforcing' efficient data transfer. We shall describe some of these features in this article.

INTRODUCTION

In this report we concentrate on how the control system protocol can be a limiting factor in scalability regarding large distributed systems. To this end it is necessary to review some popular communication strategies along with application programmer interface (API) paradigms.

DISTRIBUTED DATA FLOW

0^{th} Order: Transaction-based Client-Server

The earliest versions of most popular control system protocols made exclusive use of transaction based client-server polling. This data-flow pattern has the inherent advantage of a 'keep it simple' strategy, but can quickly run into scalability issues. These often manifest themselves as server-load problems rather than network-load problems, although both issues are important.

We take the average *load* (per second) on a server due to polling clients to be roughly given by

$$L_S \sim N_c \times N_T \times L_D \times U_T \qquad (1)$$

where L_s is the additional load on the server process due to processing client transactions, N_C is the number of clients, N_T is the average number of transactions per client, L_D is the average dispatch load of a transaction request at the server, and U_T is the average client polling rate. Equation (1) is of course schematic. The *loads* L_S and L_D will be taken to refer to the number of CPU cycles devoted to the client-side transactions.

Note that 'throwing money and threads' at the problem does *not* reduce the load as defined above. Faster, multi-core computers are of course able to do more in a given time interval. Using a thread for each transaction can also reduce the impact of sluggish servers on the client side. But in the end, the total number of CPU cycles involved will be the same (if not more, due to extra thread synchronization and context switching).

Similarly, the average load on a server's network port is

$$L_N \sim N_c \times N_T \times P_T \times U_T \qquad (2)$$

where L_N is the network load (bytes per second), N_C, N_T and U_T are as before, and P_T is the average transaction payload. This does not depend on the number of threads used or the CPU power of the server.

A real reduction in load (server or network) involves reducing either N_C and N_T or both in the above equations. This can either be accomplished artificially (for instance by imposing restrictions on the number of and location of clients allowed to run and the update rates they are allowed to use) or moving to other data flow models.

1^{st} Order: Contract-based Publish-Subscribe

As most control system data is used primarily in display at the client side, moving to an asynchronous publish-subscribe model can work wonders reducing the load on a server. Doing so eschews the 'keep it simple' approach, as connection and contract management are needed. A transaction request now results in a *contract* managed by the server, along with a table of attached clients. Nonetheless, the average load on a server due to client requests essentially becomes

$$L_S \sim N_T \times L_D \times U_T \qquad (3)$$

That is, the number of clients no longer plays a role. A transaction request is cached and made once on behalf of all N_C clients.

The outgoing network load (2) essentially remains the same, as the transaction results need to be passed to all interested parties. The incoming contribution to network load is for all practical purposes decimated, as transaction requests are made far less often. In order to further reduce the network load, one can adopt a 'send-on-change' policy, or reduce the number of clients by delivering data via multicast (especially effective for those transactions involving large payloads). The TINE control system protocol supports both of these features.

Asynchronous, publish-subscribe based protocols have a much larger domain of applicability, which however is still finite for several reasons. First, the API paradigm

still permits plaguing a server with an extra large number of transactions, N_T. Second, if application programmers have complete freedom in choosing their platforms and programming styles, client applications may still engage in synchronous polling, effectively reintroducing the N_C factor, and (depending on contract management) possibly imposing an additional asynchronous/synchronous coupling factor proportional to $(N_T)^2$.

To combat the latter two effects, one could restrict the available platforms to those *officially* approved, and to *police* the set of API standards. Or one could take steps to coerce efficient data acquisition at the protocol level. The TINE control system has now introduced many new second order hand-shaking features in this direction.

2nd Order: Contract-Negotiation

Client applications (and middle layer servers) require data from the control system for display and control of the machine. Specifically tailoring applications for efficient data transfer seldom enters into the picture. Indeed some APIs do not even offer this capability.

So on the one hand we have client applications driving control system data flow by making transaction requests (*contracts*), and on the other we have servers which bear the brunt of any ensuing scalability or efficiency problems. Servers are of course responsible for collecting the data and controlling the hardware. Thus, minimizing the impact of a server's data delivery plays a strong role regarding scalability.

Various strategies are available for reducing N_T and N_C in the above equations. In principle, one could use a purely *push* approach, where all of a server's available data are pushed via multicast onto the network. Although this might reduce the server load, it could drastically increase the overall load on the network. In addition it would require clients to sift through all data from a server in order to find the portion of interest (increasing client load). Nonetheless, pushing certain *popular* data elements (such as beam energy and current) is in general a good idea.

A server may also reduce the number of transactions it deals with if it can analyze the initial client request and, if possible, map it onto an existing contract, or anticipate further requests and appropriately restructure (*negotiate*) the contract request. We shall show below how this is done. In order to understand the principles involved, we present a brief review of control system API models.

We note that efforts to keep the dispatch load L_D to a minimum should in any case be made. The best practice involves simply copying ready data within the dispatch (rather than launching into numerical calculation or hardware readout).

CONTROL SYSTEM MODELS

Database Model

One can view the data flowing in a control system as deriving from elements in a database. This is the EPICS [3] approach, where one transfers *process variables* between the client and server. So the process variables have *names*, and the actions on the variables are one of *put*, *get*, or *monitor*.

Device Server Model

One can regard control system elements as controllable objects managed by a server. The *instance* of such an object is a *device*, which has a hierarchical name. The actions pertaining to the device are given by its *properties*. With minor differences in nomenclature and degree of object-orientation this is the model used in ACS [4], DOOCS [5], STARS [6], TANGO [7], and TINE.

Property Server Model

Certain control elements do not lend themselves well to a device oriented view but nonetheless follow the basic hierarchical naming scheme of the device server. This is typically true of middle layer services. Here one does not think of a *device* having *properties*, but of a *property* applying to different *keywords*. This model is also sometimes used in STARS and TINE, but is not available in TANGO or DOOCS.

TRANSACTION COERCION

Below we give some examples of transaction coercion and make frequent references to the *property* mentioned in the device server and property server models above, as this is the real focal point of the server transaction.

Multi-Channel Arrays

Client panels frequently attach individual elements of a *collection* to different display widgets, e.g. power supply controller (PSC) currents, beam position monitor (BPM) positions, or vacuum pressures. In large machines, this could amount to 100s if not 1000s of single elements.

TINE, however, allows a registered property to declare itself a *multi-channel array* (MCA), capable of delivering all elements of a given property as a vector (with a device order determined by the server). A rich client might directly request an MCA with all elements. Panels or strictly OO clients will not do this. However, contracts to obtain a single element of such properties are now renegotiated into a contract delivering the entire array. The client is informed via 2nd order handshaking as to the array index to device cross-reference. Thus a server only maintains a single MCA contract. The data arriving at the client is parcelled out into the individual single-element calls underneath the API. Recently, additional server side registration enables the specification of group devices, for cases where a property logically separates into sub-groups.

User-defined Types (Structures)

TINE also allows a server to define its own data types (*structures*) which a property can use in order to delivery a collection of data as an atomic unit.

Although a wonderful advent for rich client applications, structures present a display problem for

simple panel clients, which are more likely to request the structure fields individually. A server seeing such a request will deliver the entire structure, which will be re-packaged at the client. Again, many individual requests will collapse to a single contract managed by the server.

Collapsing Equivalent Contracts

In order to reduce the number of transactions it is important to make sure that equivalent calls collapse to the same contract. As aliases assume their canonical names when accessed via a client, they are unproblematic in this regard. However, a de-facto alias (device number instead of name) or an irregular array length or data type could result in a transaction occurring multiple times. Although possible to deal with via property registration, it is generally up to the server to reject non-standard requests with the appropriate error message.

Polling Intervals and Scheduling

Client applications sometimes need to know 'the moment something happens' and therefore request an update rate much faster than is otherwise necessary. A server can gracefully coerce such impatient clients to use a slower update rate by establishing a minimum polling interval. Once again, 2nd order hand-shaking renegotiates this with the client. A server can satisfy the needs of its clients by scheduling the requested property the moment there are new data to send, thereby reducing latency to essentially zero and obviating any need for fast polling.

Steering the Acquisition Mode

The payload delivered in some transactions can be very large (e.g. video frames or large traces). So even though the number of transactions might be at a minimum, the number of clients receiving the payload can result in a drain on network resources. The best practice here is to coerce all clients interested in large-payload transactions to use TINE multicast. A property can automatically renegotiate all asynchronous contracts to use multicast access (and reject synchronous requests), if so registered.

In a similar vein, properties can also reject synchronous calls in such a manner that an asynchronous listener is inserted under the synchronous call at the client side.

On the other hand, asynchronous monitoring makes no sense if the monitored data are static (do not change). An attempt to monitor such data will result in instructing the client layer to cancel the monitor.

Exclusive Read

A server can declare a property to have exclusive read characteristics, making it available only to those clients who pass through the same security screening applied to *write* transactions (*commands*). This can be used to allow time-consuming reads (e.g. extra large video signals) to be available only to a subset of the total client space.

RESULTS

Making use of these 2nd order techniques generally involves investing some time at the server front end, registering properties so that transaction coercions can take place. The benefits of doing this, however, can be dramatic. Some examples follow.

The FLASH magnet control consists of approximately 260 PSCs and is realized by various TINE servers (a primary server running on a Solaris host, and several PC104 servers running embedded linux). The client side applications are primarily DOOCS DDD [8] panels and MATLAB applications, all of which acquire settings and values from each PSC individually. Prior to introducing the techniques described above, the primary server had a constant background of ~1060 contracts, was being synchronously polled with > 500 contracts per second, and was at the high end of CPU usage. By introducing MCA access and static listeners for most of the synchronous polling, the number of background contracts is now ~ 50, there are much fewer synchronous calls, and the CPU usage is now back to 10 % or less. The client applications themselves were not modified in any way, other than relinking with the new libraries.

The mixed 100 Mbit/1Gbit infrastructure at PETRA3 introduces complications when delivering video images via multicast, especially if Gbit video servers or routers have 100Mbit video clients. As there is limited flow control, data delivery parameters must be precisely tuned. The most reliable performance was achieved by enforcing, via property registration, multicast access and a minimum polling interval.

The PETRA3 orbit server consists of ~270 Libera BPM readout modules which are attached to a single Linux CPU. Most properties are registered to provide MCA access. A minimal polling interval of 10 Hz holds the regular bevy of ~20 clients to a set of ~35 contracts and with a total CPU load of ~6 %.

We have shown in this report various methods whereby a server can take control over its clients. A server can continue to provide all callers with the information requested, but do so on its own terms.

REFERENCES

[1] http://tine.desy.de
[2] Duval et al., "TINE: An Integrated Control System for HERA", Proceedings, PCaPAC'99, 1999.
[3] http://www.aps.anl.gov/epics/
[4] http://www.cosylab.com/solutions/ICT/ACS/
[5] http://doocs.desy.de
[6] http://pfwww.kek.jp/stars/
[7] http://www.tango-controls.org
[8] http://jddd.desy.de/
[9] F.Schmidt-Foehre et al, "Control System Integration of the PETRA III BPM System based on Libera Brilliance", Proceedings, ICALEPCS 2009.

FESA3
THE NEW FRONT-END SOFTWARE FRAMEWORK AT CERN AND THE FAIR FACILITY

Alexander Schwinn, Solveigh Matthies, Dorothea Pfeiffer(GSI, Darmstadt)
Michel Arruat, Leandro Fernandez, Frank Locci, David Gomez Saavedra (CERN, Geneva)

Abstract

Currently the LHC (Large Hadron Collider, located at CERN/Switzerland) is controlled by the use of FESA2.10 (FrontEnd Software Architecture v. 2.10) classes. FESA3 is not only an update of FESA2.10, but a completely new approach. GSI plans to use the FESA system at the complex FAIR facility.

One of the main reasons to introduce FESA3 was to provide a framework which can be shared between different labs. This is accomplished by splitting up the FWK into a common part, which is used by all labs, and a lab-specific part, which allows e.g. a lab dependent implementation of the timing-system.

FESA3 is written in C++, runs a narrow interface (Remote Device Access, a middleware which encapsulates CORBA), supports multiplexing of different accelerator-cycles, is completely event driven and uses thread priorities for scheduling. It provides all FESA2.10 functionalities and additionally introduces several new features.

FESA3 is integrated in the Eclipse IDE as a plugin. Using this plugin, the user can easily create his FESA-class design (xml file), generate the C++ source code, fill the device-specific methods, and deploy the binary on a front end.

As well as the framework the Eclipse plugin has a lab specific implementation.

An operational release for FESA3 is planned end of 2010.

THE PURPOSE OF FESA3

FESA3 is a software framework which provides an easy way for developers to produce device classes by generating most of the code automatically. It supports multiplexing of different accelerator-cycles and many other features which can be used by the class-developer. The main purpose of the framework is to provide an common and unified way to develop device classes. This approach saves a lot of work and simplifies debugging, documentation and code adoption for the class-developer and all involved parties.

THE ROOTS OF THE FESA FRAMEWORK

All early versions of the FESA framework were developed solely by the CERN facility. FESA3 is the first release which is developed as an collaboration between CERN and the GSI. This collaboration was the main reason to restructure some of the Fesa2.10 fundamental internal parts and to finally go for a new major release.

FESA3 continues to provide all services from older versions and as well extends the common approach by additional services which where demanded by the CERN user community.

FESA3 AT THE FAIR FACILITY

For the FAIR facility several new accelerator installations will be built at GSI.

Central aspect is an increased number of research programs resulting in up to five beams in parallel. The FAIR facility will be controlled by a new control system which will be able to support all aspects of the complex GSI/FAIR operations on a common technical basis. The control system for the FAIR facility currently is in the design phase.

One part of this new control system will be the device software which runs on the front ends. FESA was choosen as software framework since it already proved itself at the LHC at the CERN facility and allows to pass the device-specific implementation directly to the device expert.

CLASS DEVELOPMENT WORKFLOW

The FESA3 Eclipse-plugin guides the class developer on his way to develop a FESA3 class. The following steps have to be performed to do so:

1. Design

   In the first step the developer needs to design his class according to his needs. This process involves the specification of *Properties*, *Fields*, *Server-* and *RealTime-Actions* and their dependencies on each other. The design itself is done via a comfortable XML editor, which is integrated in the FESA3 Eclipse-plugin and coupled to an XSD schema for validation.
   (see figure 1)

2. Code Generation

   Code generation may be started in the plugin if the class design is valid. An XSLT engine generates C++ code using the class design as input.

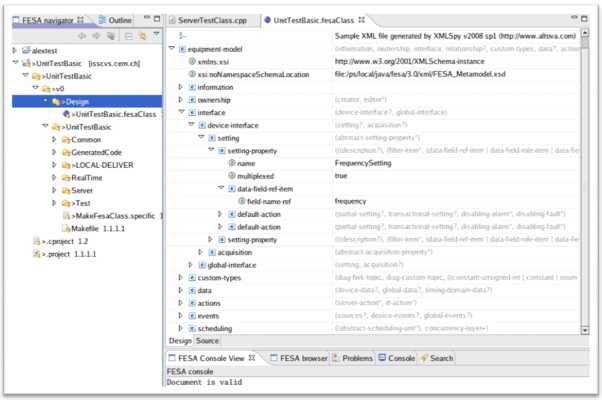


Figure 1: screenshot of the class-design in the FESA Eclipse-plugin

3. Implementation

   The code generation provides methods which need to be completed by the class developer. Those *Server- and RealTime-Actions* allow to use specific device drivers for the hardware. External driver libraries can be included in a class specific make-file.

4. Instantiation

   Since the FESA3 class may run on different front ends, the developer needs to create an instantiation document per frontend. Inside this document all configuration parameters for this specific frontend are stored. Similar to the class-design, the instantiation document is generated by the plugin and can be edited within it's xml editor.

5. Compilation

   As soon as the implementation is finished, the compilation and linking process may be started. The outcome will be a class binary for a predefined platform, which is ready to run.

6. Deployment

   The resulting binary, the instantiation file and all other dependent files need to be placed on the frontend for which they were configured and to which the device-hardware is connected.

7. Execution

   Finally the class can be executed and debugged. The FESA3 navigator-tool may help to build up a connection to the class and debug all possible scenarios.

The described workflow is not strictly forward but also allows to roll back and redo any step which is necesarry.

FUNCTIONAL OVERVIEW

Basic Internal Design

As shown by the use-case diagram below, request-handling and hardware control are the two complementary services equipment-software has to model. The two differ very much in nature since request-handling is an on-demand service, whereas hardware control is subject to tight real-time constraints. Obviously, request handling must run at a lower level of priority and shall not be able to preempt the real-time task. In order to decouple the two, equipment-software includes a software abstraction of the device. Thanks to this abstraction, an operator does not directly see the hardware device, but rather accesses it through the so called *Server side*. (see reference [1])

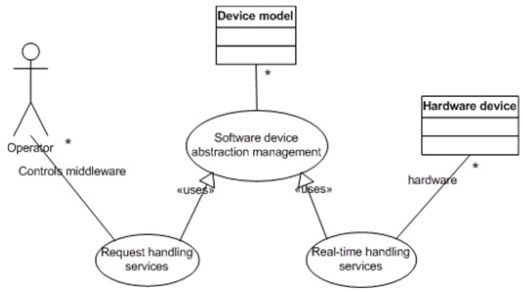


Figure 2: Separation of *Server and RealTime side*. [1]

The system is split in two logical layers: *RealTime*, which implements all parts that are directly triggered by events and *Server* which models the equipment interface and implements the middleware access. Both services are physically implemented on the same hardware platform. It is possible to run these two services in the same process, or in two separate processes. Devices are implemented as objects in the object-oriented software terminology. Each FESA3 class represents a devicetype and allows to manage different instances of this devicetype. A FESA3 equipment can represent a collection of different FESA classes, which depend on each other.

The event driven RT-system

In FESA3 *RealTime-Actions* can be triggered by different types of event, from various event sources.(see figure 3) The possible source types are listed here:

- Timing Event

  This event source is meant to be the real accelerator timing. Different events corresponding to different cycles and machines are received with this source if the frontend is connected to the timing receiver hardware.

- Timer Event

  A timer event is launched by a internal clock on a periodic interval. The developer can configure this interval in the instantiation document per frontend.

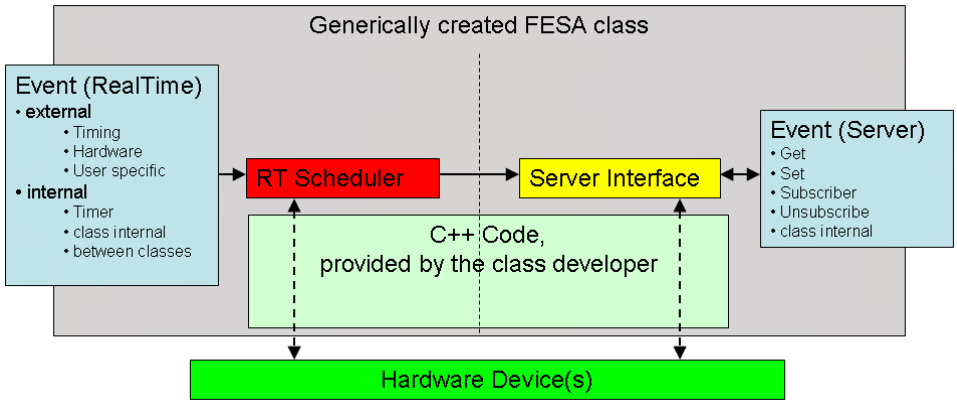


Figure 3: Basic event based functional structure of the FESA3 framework

- Custom Event

  In FESA3 the developer as well has the possibility to implement his own customized event source. This event source is used to support all other hardware and software event sources which do not use the accelerator timing.

- On Demand Event

  To communicate between *Server* and *RealTime* part of a FESA3 class there is not only the notification queue, which connects the *RealTime* to the *Server part*, but also the *On-Demand* mechanism, which works the other way around. Via socket connections it allows to trigger *RealTime-Actions* from the *Server side*.

- On Subscription Event

  FESA3 allows to establish dependencies between different FESA3 classes. E.g. one FESA3 class can subscribe to properties of another FESA3 class by using this event source.

The client interface

The client has different possibilities, to communicate with a FESA3 class using the RDA client interface. Access methods for the client are *Get*, *Set*, *MonitorOn* and *MonitorOff*. The API to these methods is a narrow one. Figure 4 shows the relation between the RDA *DeviceServer*, the middleware layer and the client.

To specify the proper attribute(s), there are several parameters:

- Property

  The property describes a collection of data, which can be obtained or modified with different client access methods.

- Device

  A FESA3 class represents a device type. A single FESA class can control many devices of the same type. Within the parameter *Device* the client can specify the proper device instance which is to access.

- CycleSelector

  On a multiplexed property, with the cycle selector string the client can select the cycle (virtual accelerator), he wants to work with. On a property which is not multiplexed the cycle selector can stay empty.

- Value

  Get methods retrieve data as an instance of the type *rdaData*. *RdaData* can store an array of mixed data types. Besides the data itself, each entry allows to store additional information.

- Context

  The context is used to pass parameters (filters) to the properties of a FESA3 class.

- ReplyHandler

  Monitor on calls require the implementation of reply handlers. As soon as new data arrives, the middleware triggers the reply handler in which the data is processed.

- Request

  This class is the virtual handle to a subscription. It is filled by the *MonitorOn* call and is used to keep track on a subscription and to terminate it if it is not needed any more.

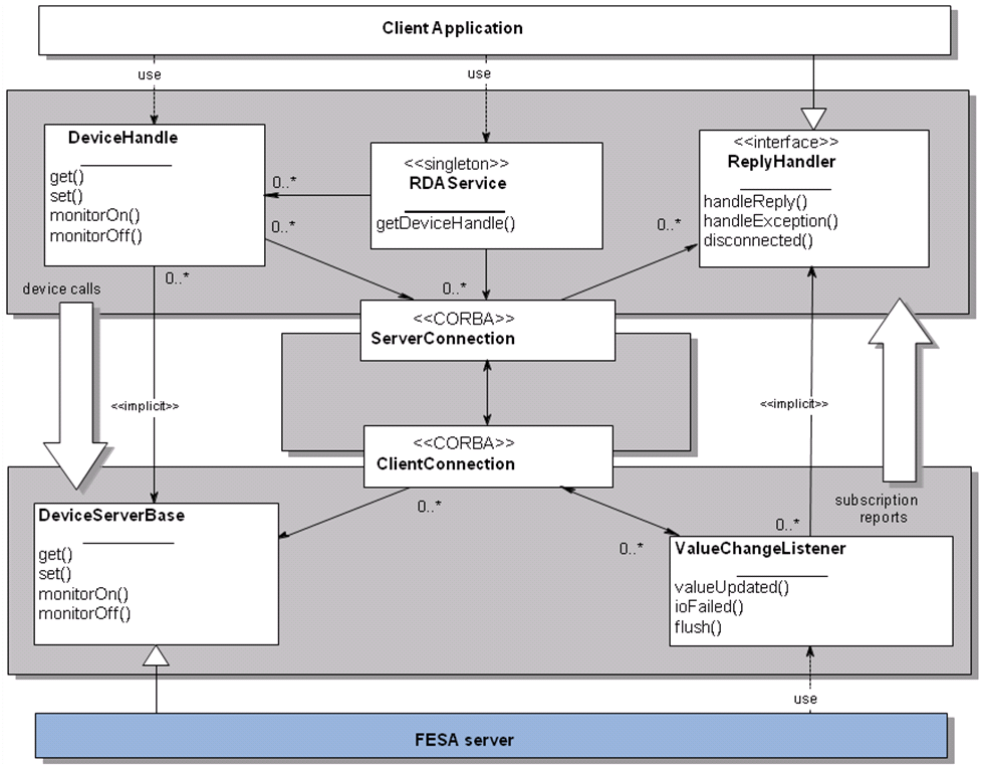


Figure 4: The CERN RDA - middleware layer. (see reference [2])

class relationship

Unlike FESA2.10, FESA3 provides three different ways of class relationship, association, composition and inheritance:

- Association (see figure 5)

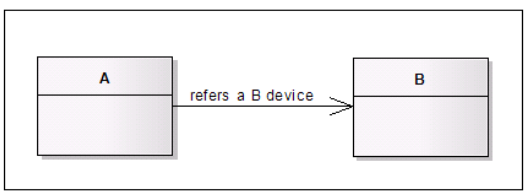


Figure 5: Association between two unique FESA3 classes

  An association specifies a light coupling between two FESA3 classes. The two classes run independently and do not rely on each other (e.g. class *A* may shut down while class *B* is still running). The two classes can be deployed on different frontends.

- Composition (see figure 6)

  Using a composition the developer can create a strong coupling between FESA3 classes. The deployment of class *A* means to deploy the whole class-tree. As well it is possible to start *B* with only *C* and *D* as sub-classes and without *A* as a smaller composition. The lifetime of the composition depends on the life-time of the base-class. All classes need to run on the same frontend. All classes within the composition are standard FESA3 classes and can be used seperately as well.

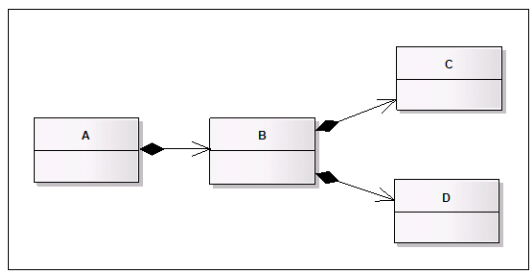


Figure 6: Composition of many FESA3 classes, represented as one class

- Inheritance (see figure 7)

  The definition of inheritance in FESA3 consists of tree characteristics:

  - *Properties/RTActions* defined by a baseclass are available for any subclass.
  - *Properties/RTActions* defined by a baseclass can be overridden (explicitly).
  - The device model of a derived class fully inherits from the device model of its baseclass.

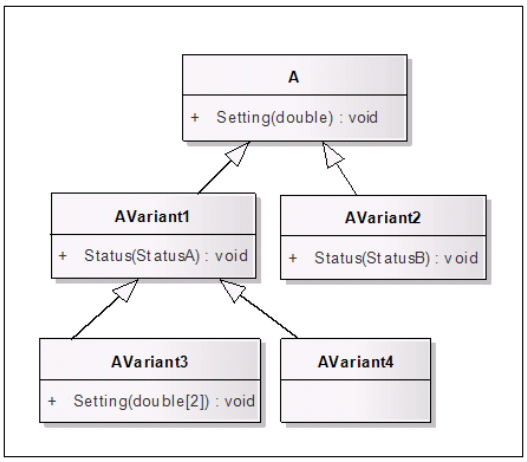


Figure 7: Inheritance between different FESA3 classes

The split between framework and lab part

The FESA2.10 framework was strongly coupled with the CERN Oracle database, the CERN Timing and several file paths. As one of the major changes this strong coupling has been removed in FESA3. Each institute which is using FESA3 now has to provide a sub-package where institute-specific code can be used. This split does not only touch the C++ code. The whole process of creating a FESA3 class is involved. Now it is possible to have a specific metamodel which triggers an adapted code generation. As well the FESA3 Eclipse plugin can be adjusted to the needs of the particular institute.

OUTLOOK

Currently FESA3 still is in the pre-beta phase. An operational beta for FESA3 will be released end of 2010. For later releases the following features and tasks are planned:

- Transaction

  On larger accelerators the possibility to synchroniously trigger a *Set* for many frontends is needed. This service is called ”transaction”.

- On-change/deadband support for subsribers

  This service allows clients to choose, if they will get notified if a value did not change at all, or if it changed only within a predefined deadband.

- Tests and benchmarks of the framework

  Tests of the FESA3 performance in terms of reaction-speed, data throughput and CPU usage need to be done.

REFERENCES

[1] A. Schwinn, D. Pfeiffer, R. Baer, “GSI-FAIR Baseline Technical Report - Front-End Software Architecture”, GSI, Germany.

[2] Kris Kostro, Joel Lauener, Nikolai Trofimov, Wojciech Gajewski, Ilya Yastrebov, “http://cmw.web.cern.ch”, CERN, Geneva

EMPLOYING RTEMS AND FPGAS FOR BEAMLINE APPLICATIONS AT THE APS*

D.M. Kline, S.K. Ross#, ANL, Argonne, IL 60439, U.S.A.

Abstract

At the Advanced Photon Source (APS), the power and flexibility of an Altera Cyclone-II FPGA combined with the Arcturus uC5282 embedded microprocessor running RTEMS, provides a low cost solution for implementing beamline applications.

In this paper, we discuss the approach of coupling an Altera FPGA and the Arcturus uC5282 to implement a time-resolved 32-channel scaler, development using the Altera Quartus-II design environment and the RTEMS tools, as well as an ASYN based EPICS device driver and its integration to the standard scaler record support. Furthermore, we discuss how this approach has been applied to other control system applications, such as for photon counting and flexible CCD shutter timing control.

By employing this approach, a variety of applications can be quickly developed on one hardware platform which realizes real-time performance within the FPGA and provide a cost effective EPICS IOC for exporting data to scientists and users.

INTRODUCTION

There exist many approaches for developing beamline and detector applications at various levels of complexity. The "Generic Digital" approach we employ abstracts the application behaviours and compartmentalizes them, serving as our "design pattern."

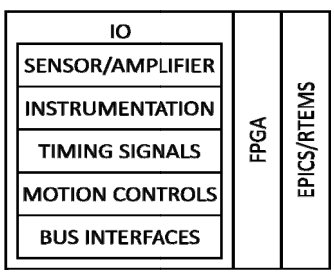


Figure 1: "Generic Digital" conceptual model.

The Input-Output (IO) component represents the specific hardware designed for the particular needs of the application. This may include inputs and outputs at different signal levels, such NIM, TTL, LVDS, or ECL. Bus adapters, such as PC/104, can be developed to take advantage of IO modules developed in-house or commercial, such as motion controllers and input sensors. Typically, the IO component connects to an in-house developed or purchased transition board, which then connects to existing infrastructure.

* Use of the Advanced Photon Source at Argonne National Laboratory was supported by the U. S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

#skross@aps.anl.gov

The FPGA component includes in-house developed base boards as well as commercially available boards, such as Altera's Cyclone or Stratix development kits. It handles the real-time aspects of the application and serves as a mediator between the IO and EPICS components.

The EPICS [1] component uses the Arcturus uC5282 [3] microcontroller as the IOC and runs the RTEMS [4] real-time operating system. It connects to the FPGA using carrier boards developed at the APS. ASYN [6] drivers are written to interface with the IO hardware. Furthermore, EPICS databases and MEDM screens have been included to implement the application behaviour and user interface.

HARDWARE

Generally, both in-house developed and commercially available hardware is employed within the model. The FPGA component uses two generations of hardware that was developed in-house. More recent versions use development kits offered by Altera.

Generation I

The generation I (GEN-I) base board consists of Altera's FLEX10K FPGA. This implementation is targeted for less complicated applications that don't require many logic elements and requires frequencies lower than 50MHz, such as "divide-by" circuits, combinational logic consisting of a few logic gates, or simple scalers.

Communication between the IOC and FPGA is through a in-house developed Serial Peripheral Interface (SPI). Typically, this is implemented on a Linux based system, such as an EPICS brick (EBRICK) [6]. The EBRICK is a Poseidon Single Board Computer offered by the Diamond Systems Corporation [5]. The FPGA is housed in a 1U rack mountable chassis and uses an external wall mount or desktop power supply.

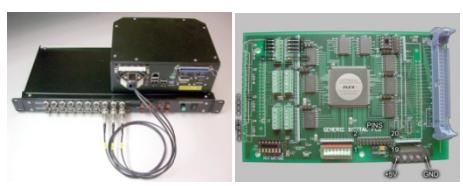


Figure 2: GEN-I/w EBRICK (left) and base board (right).

Generation II

The generation II (GEN-II) base board uses an Altera Cyclone-II FPGA [2] in a 3U VME footprint. It has IO for the uC5282, 24bits TTL IO, and 40bits LVTTL expansion IO for additional daughter boards. This implementation is targeted at higher-end applications that require more logic elements and frequencies greater than

50MHz, but less than 100MHz. Some applications include a 32-channel scaler, CCD shutter timing control, bunch intensity adjustment, and bunch photon scaler.

The uC5282 is mounted in a carrier board that was developed at the APS. EPICS is implemented on the uC5282 running the RTEMS operating system. These were chosen because of licensing, hardware and software costs, footprint, usage at the APS and in the EPICS community, known working designs, in-house expertise and support, and development environment. The FPGA base board, uC5282 carrier, and IO daughter board is housed in a 2U rack mountable chassis and use a wall mount or desktop power supply.

Figure 3: GEN-II chassis (left) and base board (right).

Altera Development Kits

Recently, development kits from Altera, which have advantages and disadvantages, are being employed. One advantage is that the development effort has already been incurred, leaving us to focus more on the application and IO component. Some disadvantages are product lifetime and the footprint, particularly when there are space constraints.

This platform is targeted for high-end applications, such as detectors, that require more logic elements and IO, frequencies greater than 100MHz, and on-board memory. In some cases, the uC5282 doesn't have enough processing power to handle the interrupt frequency, and cannot transfer the required amount of data within a short time. The FPGA handles the real-time aspects as well as storage of data to local memory. The uC5282 uploads the memory in an interleaved manner, to populate EPICS waveform records.

Figure 4: Altera Stratix-IV development kit.

SOFTWARE

Software development is split between Windows and Linux-based environments. Linux has the tools to develop EPICS, synApps [6], IOC, and the RTEMS cross-compiler. The FPGA design is developed using the Windows version of Altera's Quartus-II [7]. EPICS and synApps modules are taken from a central repository shared by APS users as well as developers from various groups. This proves to be synergistic amongst developers for support and consistency.

The IOC application resides on a central server as well, but is independent from other applications. It uses symbols derived from a script provided by synApps to reference modules and EPICS base. Our applications use the synApps EBRICK module as a departure point. "Out of the box" it supports RTEMS and the GEN-II hardware platform, including access to the uC5282 and FPGA. An ASYN driver has been written to access the common features, such as the 24bits IO. ASYN provides standard support to a variety of EPICS records. The developer doesn't need to write EPICS device support, only the driver need be written. In addition, EPICS databases and MEDM screens are available. Any specific functionality is coupled with the IOC, EPICS base and synApps modules, providing the framework and the underlying functionality required by the application.

Altera's Quartus-II design software is used to develop the FPGA logic. Interaction between the uC5282 and the FGPA is by means of the "Avalon Bus" [8]. The bus bridge, developed at the APS, mediate data transactions and interrupt processing between the uC5282 and FPGA. It gives the application visibility of the functionality provided by the FPGA and notifies it when events occur.

APPLICATIONS

The "Generic Digital" model has been applied to many applications, of varying complexity, for a number of years. Most applications use the GEN-II hardware platform which has become the "one hardware platform." The model gives flexibility allowing hardware components to be interchanged based on application complexity. It did not make sense to use GEN-II hardware when GEN-I hardware would be sufficient. Some of the GEN-II applications are discussed below.

One of these applications is a 32-channel scaler. Although commercial ones are available, there were several reasons for developing one in-house, such as lower hardware cost, ability to deploy them in beamlines without VME, and mobility between beamlines.

The primary requirement of the scaler was to behave identically to existing systems, and make use of current EPICS support and infrastructure. The GEN-II hardware was used and an IO board was developed to level shift the LVTTL to TTL. A breakout board with Lemo connectors and LED indicators was developed that mounts to the front panel and connects to the IO board with a flat ribbon cable.

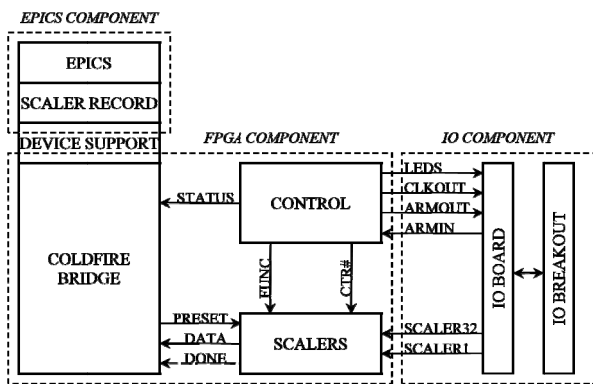


Figure 5: Scaler block diagram.

The front panel consists of Lemo connectors, a power ON/OFF switch and indicator. There is input for each scaler, and an ARMIN and GATEIN for daisy chaining with external equipment. There is a 10MHz CLKOUT output for external synchronization, and an ARMOUT output that is controllable through EPICS. Typically, for time resolved applications, the ARMOUT is connected to the ARMIN and the CLKOUT to the first scaler.

The back panel has a power input connector. Typically, an external wall mount or desktop power supply is used for electrical safety concerns. There are LEDs to indicate activity from EPICS, ARMED, ARMIN, and GATEIN.

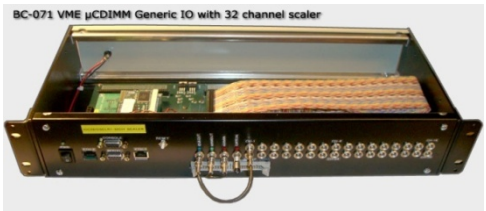


Figure 6: Scaler unit.

Most of the EPICS components were already written. EPICS databases, MEDM screens, and scaler record support were taken from the STD synApps module [6]. An ASYN driver was written to provide device support and define interfaces that allow PVs access to the scalers outside of the record context. Record support provides a framework which calls methods from the ASYN driver in a programmed sequence depending on its counting mode. The driver responds by sending commands and read/write data through the bridge to control the scalers.

The driver defines methods for initializing hardware and scalers, reporting its status, processing scaler values, and handling interrupts from the FPGA. During initialization, the driver registers its methods with the record using the Device Support Entry Table (DSET) structure [9] hooking them into the framework. An interrupt handler is registered with RTEMS to process status and read out scalers, and post them to EPICS.

The FPGA component defines Programmed IO (PIO) registers which hook into the bridge. Registers are accessed by the ASYN driver as memory locations. Commands and data received by the driver indicate what function to perform, such as arm, preset, or read a scaler. All scalers are up counters. Preset values are converted to negative prior to loading into the scaler. When a scaler reaches zero, it generates an interrupt to the driver and dispatches it to a process which posts the status and data to EPICS.

Similar to the scaler, other applications have been developed which use the GEN-II hardware platform. A flexible timing module was developed for CCD shutter controls. It allows the user to delay and stretch timing signals from field instrumentation with 20nS resolution. For this application, no new hardware needed to be developed. A “bunch scaler” was developed to count the number of photons per accelerator bunch from an Avalanche Photo Diode (APD) for a Laser-based pump probe experiment. A new IO component board was required because the field equipment had intermixed signal levels, such as TTL, ECL, and NIM. Software development for both these applications only required an ASYN driver and MEDM screens for the application specific behaviour and user interfaces.

CONCLUSION

The “Generic Digital” approach provides a design pattern that can be employed to develop and rapidly deploy many beamline and detector applications. The model allows flexibility and the ability to adapt to applications of varying configurations and complexities. Existing hardware components can be easily interchanged and new ones developed. Coupling the uC5282 with an FPGA is a hardware configuration that has been proven to be reliable at the APS and other laboratories throughout the community. Using EPICS, RTEMS, and synApps, reduces overall project cost and allows one to focus more on the application development, thus minimizing the hardware and software development time. Both future beamline and detector applications and those currently under development, along with the APS users, will benefit by the approach discussed in this paper.

REFERENCES

[1] EPICS; http://www.aps.anl.gov/epics.
[2] Altera Cyclone-II Field Programmable Gate Array; http://www.altera.com/products/devices/cyclone2.
[3] Arcturus Neworks, uC5282 microprocessor module; http://www.arcturusnetworks.com/products/uc5282/.
[4] RTEMS; http://www.rtems.org.
[5] Diamond Systems; http://www.diamondsystems.com.
[6] synApps; http://www.aps.anl.gov/bcda/synApps.
[7] Altera Quartus-II Development Software Suite; http://www.altera.com/literature/lit-qts.jsp.
[8] Avalon bus memory mapped interface specification; http://www.altera.com/literature/lit-sop.jsp.
[9] M.R. Kraimer, et al., “EPICS Application Developers Guide,” February 2010, p. 170 (2010); http://www.aps.anl.gov/epics.

QT EPICS DEVELOPMENT FRAMEWORK*

A. Rhyder#, A. Owen, G Jackson. Australian Synchrotron

Abstract

QCa is a layered software framework based on Qt for accessing EPICS data using Channel Access on a range of platforms. It is used on several beamlines at the Australian Synchrotron. The QCa framework provides object oriented C++ access to control systems using EPICS (Experimental Physics and Industrial Control System). It is based on Qt, a widely used cross-platform application development framework. GUI or console based applications can be written that use QCa at several levels. QCa includes Qt plugin libraries, EPICS aware widgets, data formatting classes, and classes for accessing raw EPICS data in a Qt friendly way. QCa also includes an application for displaying forms produced by the Qt development tool 'Designer'. Using this application a complete EPICS GUI system can be generated without writing any code. A GUI system produced in this way can interact with existing EPICS display tools such as EDM. QCa handles much of the complexities of Channel Access including initiating and managing a channel. Applications using QCa can interact with Channel Access using Qt based classes and data types. Channel Access updates are delivered using Qt's signals and slots mechanism.

INTRODUCTION

Channel Access is described as 'one of the core components of an EPICS system. It is the software component that that allows a Channel Access client application to access control-system data which may be located on different hosts throughout a network' [1]

While CA is the default means to access EPICS data, its use is not trivial. A significant understanding of how CA works is required to execute the steps required to read or write data. The complexity of setting up and terminating CA requests leaves room for error. Also, CA uses a C programming interface and so does not make use of object oriented programming techniques.

Qt is a cross-platform application and UI framework. It includes a C++ class library and a cross-platform IDE.

The QCa framework provides a Qt based C++ framework for easy CA access to EPICS data.

It provides access to EPICS data at several levels from programmatic reading and writing of data, EPICS aware widgets for developing GUI based applications through to EPICS aware Qt plugins such as push buttons, sliders, and text widgets. When these plugins are used within the Qt form development tool 'designer' EPICS GUIs can be developed without the need for any code development.

QCA FRAMEWORK HIERARCHY OVERVIEW

The QCa framework is designed to allow access to CA data in the most appropriate form. The framework is based on a hierarchy of classes as shown in Table 1. This hierarchy is open at all levels to the developer. Appropriate use of the hierarchy is shown in Table 1.

Table 1: QCa framework hierarchy

Type of access to CA data.	**Functionality**	**Main classes**
C++ access to the CA library.	Provides convenient C++ access to the CA library.	CaObject
Qt based access to CA.	Hides CA specific functionality. Adds Qt functionality such as signals and slots.	QCaObject
Data type independent access.	Hides EPICS data types, providing read and write conversions where required.	QCaInteger QCaString QCaFloating
EPICS aware graphical widgets.	Implements graphical Qt based widgets that provide access to EPICS data.	QCaLabel QCaLineEdit QCaPushButton QCaShape QCaSlider QCaSpinBox QCaComboBox QCaPlot
EPICS aware graphical Qt plugins.	Adds Qt plugin interfaces to EPICS aware widgets.	QCaLabelPlugin QCaLineEditPlugin QCaPushButtonPlugin QCaShapePlugin QCaSliderPlugin QCaSpinBoxPlugin QCaComboBoxPlugin QCaPlotPlugin
GUI support widgets	Implements Qt based widgets that support control system GUIs. These widgets do not access the CA library.	AsGuiForm GuiPushButton CmdPushButton Link

*Work supported by the Australian Synchrotron
#andrew.rhyder@synchrotron.org.au

C++ ACCESS TO THE CA LIBRARY

The CaObject base class provides a C++ wrapper around the CA library. While available to the developer, it was written mainly to provide a level of abstraction within the Qt based QCaObject class. It is recommended to be used where a Qt framework is not available.

QT BASED ACCESS TO CA

The QcaObject class provides full access to EPICS data while hiding most CA specific functionality such as link status, connections and channels.

The QcaObject class adds Qt functionality. Data can be written using a Qt slot and Qt signals are available for data and status information as required. Qt data types are used to represent all EPICS data.

The data in the update signals may be of any type and is represented by a Qt variant.

DATA TYPE INDEPENDENT ACCESS

The classes QCaInteger, QCaString, and QCaFloating are based on QCaObject and interpret all data as integers, strings, and floating point numbers respectively. They are used to provide access to EPICS data in a known format regardless of the actual data type of the EPICS data. For example, string data is always required for a text label regardless of the underlying EPICS data type. While some conversions are unlikely to be of much practical use, all conversions are permitted.

EPICS AWARE GRAPHICAL WIDGETS

The classes QCaLabel, QCaLineEdit, QCaPushButton, QCaShape, QCaSlider, QCaSpinBox, QCaComboBox, and QCaPlot allow an application to add graphical objects to a user interface that are EPICS aware. That is, they interact directly with EPICS data. The application sets up the EPICS process variable name and other parameters that define how the widget interacts with EPICS data. The application does not have to handle EPICS data or any aspect of the CA interface.

The application may supply the EPICS aware widgets with an object that the widgets can send Qt signals to, including error and status messages signals.

EPICS AWARE GRAPHICAL QT PLUGINS

The classes QCaLabelPlugin, QCaLineEditPlugin, QCaPushButtonPlugin, QCaShapePlugin, QCaSliderPlugin, QCaSpinBoxPlugin, QCaComboBoxPlugin, and QCaPlotPlugin are EPICS aware widgets with a Qt plugin interface.

These plugins can be used by any Qt application that can load plugins.

They are loaded into the Qt GUI design tool 'Designer' which can be used to generate GUI description files that include EPICS aware widgets. These files can be loaded at run time by any application code, or used as source for any application. One application that loads these files at run time is AsGui, an MEDM/EDM replacement. A feature of these plugins is that they are active at design time.

GUI SUPPORT WIDGETS

The classes AsGuiForm, GuiPushButton, CmdPushButton and Link implement Qt based widgets that support the development of EPICS control system GUIs. They are not EPICS aware widgets.

The AsGuiForm class can contain any Qt based widgets, including the QCa framework's widgets. It is used as the scroll area in the AsGui application and can be used to create sub forms when developing control system GUIs in 'designer'.

The GuiPushButton class is used to launch new GUIs.

The CmdPushButton class is used to execute any command. Typically it would be used within a GUI to perform an action on the local machine, such as launch another application, or interact with an MEDM session.

The Link class provides a generic mechanism for configuring how widgets in a GUI interact. For example, the value in one widget can control the visibility of another. Examples of the GUI support for Qt plugins are shown in Figure 1.

QCA BASED APPLICATIONS

The QCa framework currently includes a couple of applications. The main application is AsGui.

AsGui is a graphical control system user interface. It displays EPICS aware GUIs based on user interface files created using 'Designer' as shown in Figure 1.

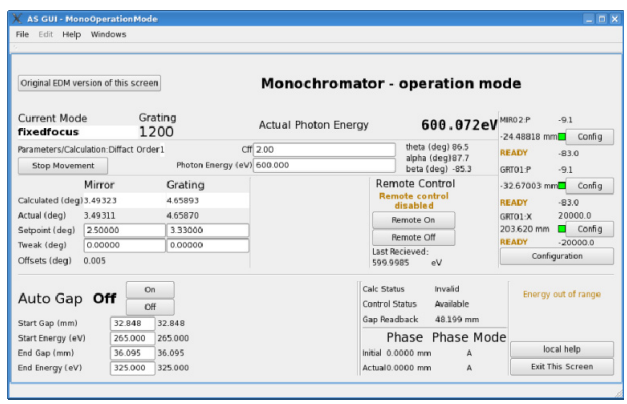


Figure 1: A sample GUI created in designer using EPICS aware plugins and GUI support plugins

QCaMonitor is a console application that takes a list of EPICS process variable names as an argument and monitors changes to the data specified by the names. It will perform the same task as the standard EPICS application caget. It is an example of using QCaString objects to generate a stream of textual based updates.

CLASS USAGE

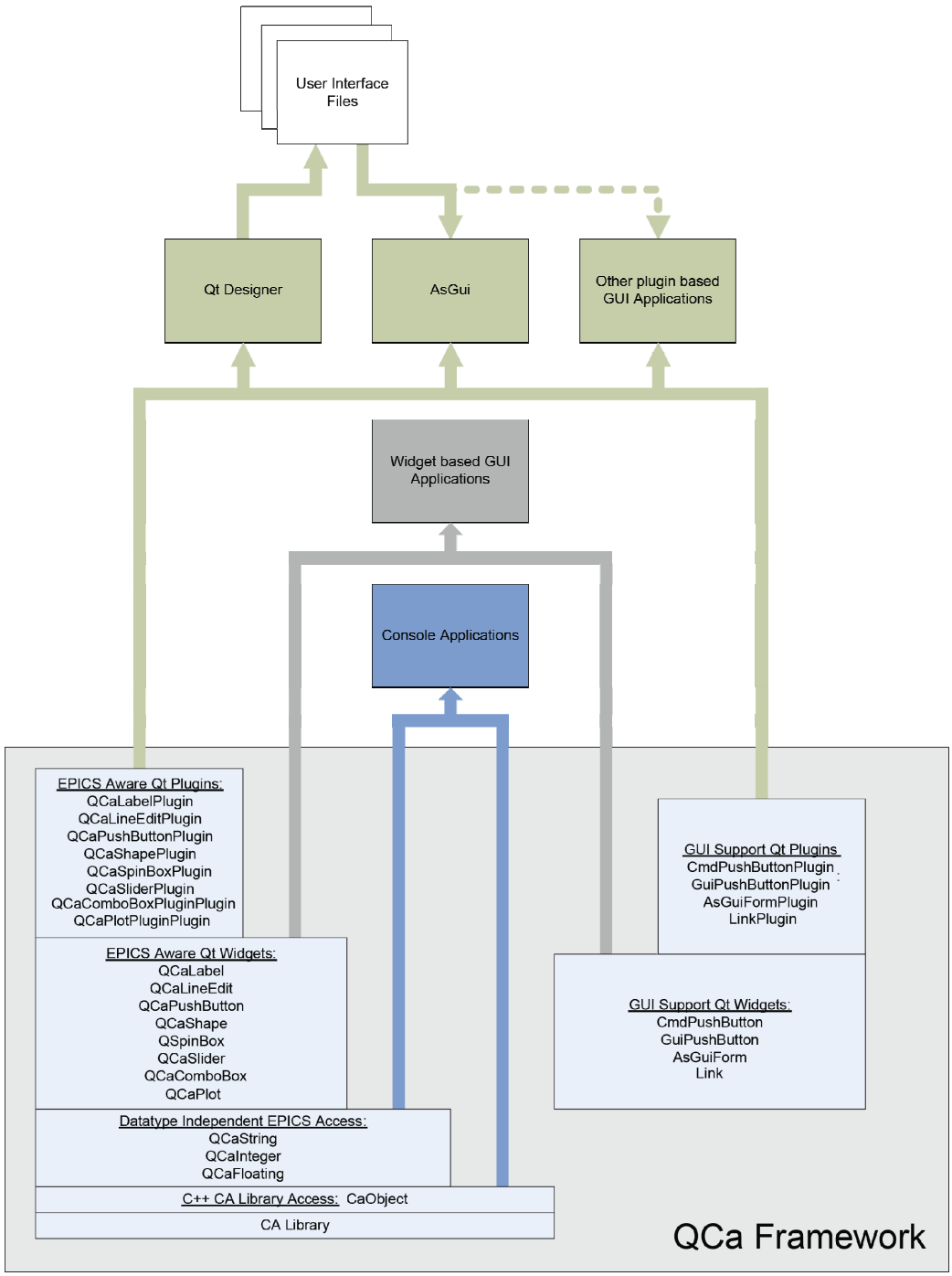


Figure 2: Typical QCa class usage

REFERENCES

[1] Philip Stanley Channel Access Client Library Tutorial. Los Alamos National Laboratory. http://lansce.lanl.gov/EPICSdata/ca/client/caX5Ftutor-1.html

THE BEAMLINE EXPERIMENTS SCHEDULING SOFTWARE*

Yuhong Yan#, Ludeng Zhao, Zhiguo Wang, Yongxin Zhu, and Chun Wang,
ENCS, Concordia University, Montreal, QC H3G1M8, Canada

Abstract

Scheduling the experiments to the beamlines of the synchrotron at the Canadian Light Source (CLS) is a manual procedure so far. Once every six months, the beamline scientists discuss before a whiteboard to schedule as many approved experiments as possible. There are so many constraints on resource capabilities, availabilities, user preferences, and experiment priorities to consider that none has ever been able to check if the manual scheduling results are optimal or not. In the Canarie funded project Science Studio, we are building an automatic scheduling module as a part of the User Office. After the synchrotron users submit their proposals via the User Office, the automatic scheduling module can find an optimal scheduling solution that satisfies all the constraints modelled, if such a solution exists, and display the results on a Web calendar. In this paper, we present our contributions on design and implementation of the scheduling module and our study on automatic scheduling of synchrotron experiments.

THE BACKGROUND

The automation of the scheduling activities at the CLS is part of the Canarie funded project Science Studio. The Science Studio project develops a complete experiment management system [1] that allows the researchers to control the experiment devices, observe the experiment processes, and collect data from their own home bases, instead of travelling to the CLS site.

There are about 30 plus the CLS like facilitates [2] around the world. All the facilities have similar proposal approval procedures, regardless the different frequencies of calls-for-proposals and the length of minimal time slot. Scheduling the approved proposals is done manually. In the CLS, the beamline scientists who are in charge of scheduling experiments on the beamlines use e-mail and documents like spreadsheet and pdf as their primary tools to communicate with the users and manually scratch the schedules on a calendar. In order to make their lives easier, the beamline scientists tend to limit the possible combinations they should consider. Furthermore scheduling under conflicting constraints can easily become intractable as the number of users and proposals increase.

In this paper, we present our solution to automate the scheduling function. The User Office in the Science Studio platform has a proposal management module to manage the proposal submission and review procedure. In the CLS, the call-for-proposals occurs every six months (*aka*. a cycle). The approved proposals can be scheduled into the next four cycles. Our schedule module reads in the proposal information, invokes the scheduling tool, and displays the scheduling results a Web calendar. The beamline scientists can review the schedule and manually fine-tune the schedule over the Web calendar. The design and the model of experiment scheduling can be reused for different facilitates as well. The parameters of the scheduling model and the Web UI of the implementation are tuned for the CLS.

THE SYSTEM ARCHITECTURE

The Science Studio platform is a large J2EE enabled Web application. Figure 1 shows its system architecture. The core of the system architecture is the application tier composed by the User Interface (UI) services, the User Office, and the beamline services.

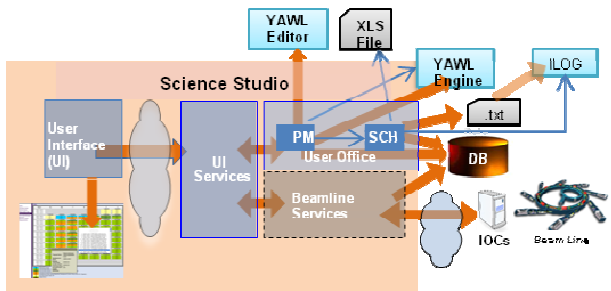


Figure 1: System architecture for Science Studio.

The major functions of the User Office are proposal management (PM in Fig. 1) and scheduling (SCH in Fig. 1). The proposal management module accepts user inputs and manages the proposal review process. The proposal review process can be executed by a workflow engine YAWL. YAWL Editor is used offline to design the proposal review process. When the approved proposals are decided, the scheduling function is invoked.

The scheduling module gets the approved proposals from the database or from the excel sheets currently used, and converts the information about the defined experiments into a text format that can be accepted by the ILOG - the automatic scheduler. A .txt file containing the converted data is the media of transferring the data between Science Studio and the ILOG. The ILOG is invoked by the scheduling module. The ILOG writes the scheduling results into a text file. The scheduling module reads the results and shows them on the Web calendar, meanwhile stores them into the database.

*Work supported by the Science Studio project which is funded by Canarie Network Enabled Platform Program contract number NEP-01
#yuhong@encs.concordia.ca

The UI service generates the Web UI. The users interact with the system through a common Web browser.

The beamline services offer the functions to observe the experiments and operate the physical devices from a Web browser. The underlying beamline control system is implemented using EPICS based Input/Out Controllers (IOCs) and with report access capability. This module is not the focus of this paper.

MODELING THE SCHEDULING PROBLEM

The experiment scheduling problem is modelled as an integer programming model.

Suppose the CLS has *m* beamlines $I = \{1, \ldots, m\}$ and *n* approved proposals $J = \{1, \ldots, n\}$. Currently a proposal defines one experiment. So we use experiment and proposal interchangeably in the following text. Experiments and beamlines are characterized by the following parameters which represent the clients' preferences and the scheduling constrains:

The cycle start time *St* defines when the synchrotron scheduling cycle starts. A cycle is six months, e.g. from 2010/1/1, 0:00AM to 2010/6/30, 12:00AM. Therefore, *St* can be the time point of 2010/1/1, 0:00AM.

The cycle end time *Se* defines when the synchrotron scheduling cycle ends.

The experiment unacceptable start time *Us*[*j*, *o*], where $j \in J$, and $o \in \{1, \ldots, Max1\}$, indicates when the *o*-th unacceptable period starts for the experiment *j*. *Max1* is a constant that an experiment can define up to *Max1* unacceptable periods. In the CLS, *Max1* = 6.

The experiment unacceptable end time *Ue*[*j*, *o*], where $j \in J, o \in \{1, \ldots, Max1\}$ indicates when the *o*-th unacceptable period ends for the experiment *j*.

The experiment release time *R*[*j*], where $j \in J$, is the earliest possible start time of the experiment *j*. Before that time, the experiment *j* cannot be scheduled.

The preferred end time *D*[*j*] where $j \in J$ is the latest preferred finishing time for the experiment *j*. The experiment *j* should be scheduled before this time.

The processing time *P*[*j*], where $j \in J$, is the time duration to complete the experiment *j*.

The weight *W*[*j*], where $j \in J$, represents the priority given to the experiment *j*. Many factors can determine the priority for an experiment. For example, the proposals with biological samples have higher priority, and the commercial proposals have higher priority than the normal academic proposals.

The eligibility $E[i,j] \in \{0,1\}$, where $i \in I, j \in J$, is a Boolean value. When *E*[*i*,*j*] = 1, the experiment *j* can be conducted on the beamline *i*.

The beamline has some down time that is unusable for experiments. The beamline unusable start time *Ub*[*i*,*u*], where $i \in I$, $u \in \{1, \ldots, Max2\}$, indicates when the *u*-th unusable period starts. *Max2* is a constant for the maximal number of unusable periods. The beamline unusable end time *Ua*[*i*,*u*], where $i \in I$, $u \in \{1, \ldots, Max2\}$, indicates when the unusable period ends.

The above variables contain the known facts of our model. The following variables are going to be assigned their values by the scheduling algorithm.

The experiment start time *S*[*j*], where $j \in J$, is the scheduled time for starting the experiment *j*.

The assignment $X[i,j] \in \{0,1\}$, where $i \in I, j \in J$, is a Boolean value. *X*[*i*,*j*]=1means the experiment *j* is assigned to the beamline *i*.

The scheduling has to adhere to the following rules:

Only the eligible beamlines can be selected:

$$\forall i,j, X[i,j] \le E[i,j] \quad (1)$$

One beamline per experiment:

$$\forall j, \sum_{i=1}^{m} X[i,j] = 1 \text{ s.t. } X[i,j] \in \{0,1\} \quad (2)$$

The experiment start time should be greater or equal to the release time:

$$\forall j, S[j] \ge R[j] \quad (3)$$

The experiment start time should be greater or equal to the cycle start time:

$$\forall j, S[j] \ge St \quad (4)$$

The experiment end time should be less or equal to the cycle end time:

$$\forall j, S[j] + P[j] \le Se \quad (5)$$

On a beamline, the experiments can't overlap:

$$\forall i,j,k, \text{s.t.} j \ne k, k \in J$$
$$S[i,j] \ge S[i,k] + P[k] \vee S[i,k] \ge S[i,j] + P[j] \quad (6)$$

An experiment should be out of the unacceptable time window:

$$\forall i,j,u, S[i,j] + P[j] \le \text{Ub[i,u]} \vee S[i,j] \ge Ua[i,u] \quad (7)$$

An experiment can only be arranged in the beamline available period:

$$\forall j,o, S[j] + P[j] \le Us[j,o] \vee S[j] \ge Ue[j,o] \quad (8)$$

The objective of the problem is to minimize the total weighted lateness which is defined as the sum of time differences between the preferred end time of an experiment and its actual finish time. It is a criterion representing how much the schedule satisfies users' expectations in terms of users' preferred finish times.

$$\sum_{j=1}^{J} |W[j] * (S[j] + P(j) - D(j))| \quad (9)$$

SOFTWARE IMPLEMENTATION

The Web UI for the Calendar

The Science Studio platform has a common Web UI for all its modules (Figure 2). The left vertical bar shows the menu items, and the current item is the "automated schedule". A calendar in the content pane shows the schedule for a beamline.

The color encoding represents the different operation modes of the beamlines. For example, the normal mode is in green. Our scheduling application is implemented for two purposes: first, it supports manual scheduling by providing the beamline scientists with a calendar liked interface, on which the beamline scientists can define their facility operation modes and manually schedule eligible experiments onto the shifts; second, the scheduling application is able to invoke the automated algorithm resides the in ILOG and retrieve the results back for displaying them on its calendar UI. Figure 3 shows some of the manual operations. With a right click on a time slot,

a drop-down list pops up, containing all the schedulable experiments. The user can check the detail information of an experiment by right-clicking the experiment in the list.

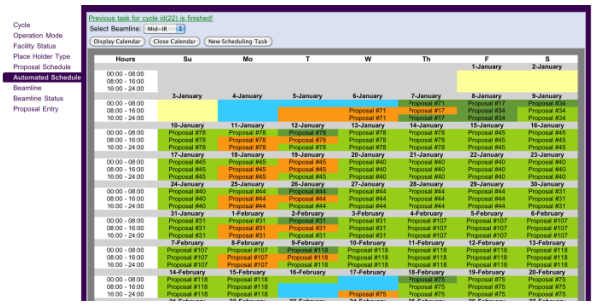


Figure 2: Example screen shot showing a calendar with schedule results.

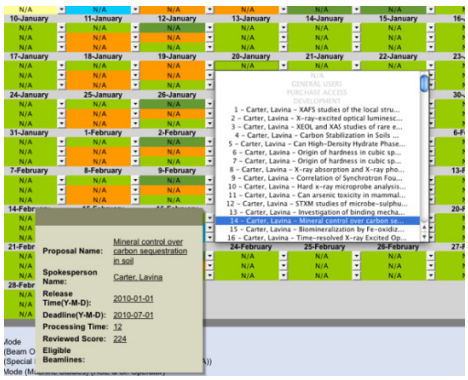


Figure 3: Example screen shot (enlarged) for illustrating some manual scheduling operations.

Testing with the real world data

We use the CLS proposals data for the first cycle of 2010 for testing the scheduling functions. A summary of these proposals is in Table 1. Totally 141 proposals by about 100 spokespersons from about 50 institutes are approved, most of which are from Canadian institutes.

Table 1: General user proposal summary

Beamline	Total Requests	Total Shift Request
01B1-1 (Mid IR)	5	77
02B1-1 (Far IR)	15	258
06B1-1 (SXRMB)	9	72
06ID-1 (HXMA)	25	211
07B2-1 (VESPERS)	5	60
10ID-1 (SM)	36	343
11ID-1 (SGM)	32	228
11ID-2 (PGM)	14	120
Total	**141**	**1369**

These proposals are manually scheduled to 8 beamlines. Each row in Table 1 shows the number of the proposals scheduled on one beamline and the number of shifts used. We show we can do the same with the automatic schedule function in our module.

Figure 4 shows the input data file for the ILOG. We convert the shifts into nature numbers. For example, the first shift is from 2010/1/1, 0:00am to 2010/1/1, 8:00am, and it is converted to 1. In Figure 4, *M*=8 is the number of the beamlines. *N*=141 is the number of experiments to be scheduled. *R* is the array of release time for all the experiments. As all the experiments have the release time from the first shift in the cycle, all the items in *R* are 1. *W* is the array of the weights of the proposals. Each of numbers in *W* is a proposal review score multiplied by 100. *P* is the array of processing time. *E* is an 8×141-array of the eligibilities for the 141proposals on 8 beamlines. We can see the first 5 proposals are eligible for beamline 1. Variable *Ub* and *Ua* are 8×30-array of unusable start time and end time. 30 is the value for *Max2* obtained from the real data. Variable *Ts* and *Te* represent the cycle start and end time. The available shifts are between *Ts* and *Te* except pairs of *Ub* and *Ua*. As the first numbers of Ub and Ua are 1 and 14, shifts 1 to 13 are unusable.

Figure 5 shows the schedule results in a text file outputted by the ILOG. Each item is in the format of [*the index of beamline, the index of proposal, the allocated start time, the allocated shifts*]. For example, the first item [7, 122, 14, 1] means on beamline 7 (11ID-1), the 122-th experiment is scheduled from the 14-th shift for 1 shift.

```
M=8;
N=141;
R=[1,1,1,1,1,...];
W=[171,200,229,291,300,...];
P=[24,8,9,21,15,...];
E=[[ 1,1,1,1,1,...],[ 0,0,0,0,0,1,...], ...];
Ub=[[ 1,16,140,242,340,343,431,542,...],...];
Ua=[[14,17,144,338,341,344,435,544,...],...];
Ts=1;
Te= 544;
Us=[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],...];
Ue=[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],...];
```

Figure 4: Example of the input data for the ILOG.

```
7,122,14,1    7,110,42,6    7,106, 98,9    7,123,189,6
7,118,17,3    7,111,48,6    7,102,107,9    7,113,203,9
7,100,20,3    7,112,54,6    7, 96,116,10   7,108,212,9
7,103,23,3    7,116,60,6    7,119,126,12   7,107,221,15
7,109,26,3    7,105,66,8    7,125,138,2    7,121,236,6
7,124,29,3    7,126,74,8    7,127,144,15   7,101,521,6
7, 97,32,4    7,114,82,8    7, 98,159,18   7,104,527,6
7,120,36,6    7, 99,90,8    7,115,177,6    7,117,533,9
```

Figure 5: Part of the scheduling results from the ILOG for beamline SGM.

REFERENCES

[1] Canadian Light Source, Remote Instrument Control, http://www.lightsource.ca/operations/remote_access_project/

[2] Wikipedia Article, Synchrotron, Wikipedia, the free encyclopedia. en.wikipedia.org/wiki/Synchrotron. March, 2009.

[3] Zahid Anwar, Zhiguo Wang, Chun Wang, Dan Ni, Yaofeng Xu, Yuhong Yan, “A Integer Programming Model and Heuristic Algorithm for Automatic Scheduling in Synchrotron Facilities”, 2009 IEEE Inter. Conf. on Systems, Man, and Cybernetics (SMC'09), Oct. 11-14, 2009, San Antonio, Texas, USA.

ACCURATE MEASUREMENT OF THE BEAM ENERGY IN THE CLS STORAGE RING*

J.M. Vogt#, J.C. Bergstrom, S. Hu, CLS, Saskatoon, Canada

Abstract

Resonant spin depolarization was used at the Canadian Light Source (CLS) to measure the energy of the beam in the storage ring with high accuracy. This method has been employed successfully at several other synchrotrons in the past. At the Canadian Light Source, however, resonant spin depolarization is an intrinsic capability of the transverse feedback system, which is based on a Libera Bunch-by-Bunch unit. The Bunch-by-Bunch system used at the CLS was customized to include a bunch cleaning feature based on a frequency-modulated oscillator. By setting the frequency of this oscillator to the spin tune, the beam can be depolarized and the effect can be observed by watching the life time of the beam. No changes have to be made to the permanent setup of the transverse feedback system, and no special instrumentation is required to make the energy measurement.

RESONANT SPIN DEPOLARIZATION

The theory of resonant spin depolarization as a means of measuring the beam energy in a storage ring has been described in detail in Ref. [1]. After injection, the beam polarization builds up with a machine-dependent time constant, usually in the range of a few tens of minutes. Depolarization is then accomplished by applying an RF-signal at the resonant frequency of the spin. The effect of the resonant depolarization is observed either as an increase in the amount of Touschek scattering, or as a decrease of the beam life time. Several facilities have used this method in the past [1-7].

The frequency at which resonant depolarization occurs is a direct measure of the beam energy. Equation (49) in Ref. [1] gives the spin tune ν as:

$$\nu = a\gamma = a\frac{E}{m_e c^2}, \qquad (1)$$

where

$$a = \frac{g-2}{2} = 0.00115965$$

is the anomalous magnetic moment of the electron, E is the beam energy, and m_e is the electron mass. At the nominal beam energy of the CLS storage ring, which is 2900 MeV, the spin tune is ν = 6.5812.

The expected resonant depolarizing frequency f_{dep} is:

$$f_{dep} = \nu_{frac} \cdot f_o = 1.0197\,\text{MHz}, \qquad (2)$$

where ν_{frac} is the fractional part of the tune and f_o = 1.7544 MHz is the orbit frequency of the storage ring. Note that there is an ambiguity between $\nu_{frac} = 0.5812$ and $1 - \nu_{frac} = 0.4188$, so that another solution for the depolarizing frequency is:

$$f_{dep} = (1 - \nu_{frac}) \cdot f_o = 0.7347\,\text{MHz}. \qquad (3)$$

INSTRUMENTATION AT THE CLS

The Transverse Feedback System

The transverse feedback system is based on a Libera Bunch-by-Bunch unit, which was customized to include a frequency modulated oscillator for bunch cleaning [8]. The frequency of this oscillator was set to the spin tune and the amplifiers and the vertical kicker of the transverse feedback system were used to depolarize the beam.

Detection of Depolarization

Because of signal-to-noise considerations, the preferred method of detecting depolarization is by measuring Touschek electrons. However, the arrangement of the magnets in the storage ring and the shape of the vacuum chambers make it impossible to set up Touschek detectors at the CLS. Depolarization therefore had to be detected by observing its effect on the life time of the beam.

MEASUREMENTS

Machine Setup

The machine setup was determined by the following considerations:

- In order to maximize the Touschek effect on the life time, the bunch current had to be as high as possible,
- The bunch current was limited by the head-tail instability,
- In order to minimize the vacuum effect on the life time, the total current had to be as low as possible,
- The total current had to be high enough for a sufficiently accurate measurement of the storage ring current and the life time.

As a compromise, three bunches in the storage ring were filled with a current of about 10 mA/bunch.

*Work supported by NSERC, NRC, CIHR, WEDC.
#johannes.vogt@lightsource.ca

Frequency Sweep

The frequency-modulated oscillator was swept in a range of frequencies that included the expected resonant depolarization frequency. The product of life time and beam current was observed (see Fig. 1). If the life time of the beam is only determined by Touschek scattering, this product is expected to be a constant as long as the polarization of the beam does not change. When the beam is depolarized, Touschek scattering increases and the product of life time and beam current is expected to drop. In reality the product increases slowly, probably due to a contribution to the life time by the vacuum in the storage ring, which slowly improves as the beam current decays.

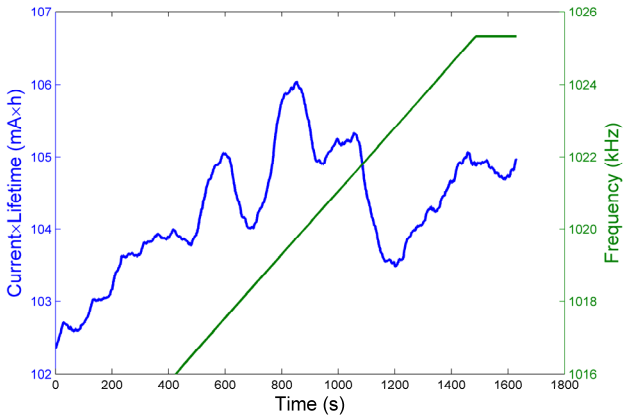


Figure 1: The blue curve shows the product of beam current and life time. The green curve is the frequency of the oscillator. The blue curve drops between t=800s and t=1200s as the beam is depolarized.

Because of the fluctuation of the current × life time measurement, the depolarization frequency could not be read with the desired accuracy. The measurement was therefore repeated several times after the beam was allowed to polarize again, and each time the range of the frequency sweep was narrowed. In the end the sweep was made narrower than the range necessitated by the energy spread of the beam in the storage ring, and the beam was partially depolarized.

RESULTS

The depolarization frequency determined in this manner was

$$f_{dep} = 1.019 \pm 0.001\,\mathrm{MHz}\,.$$

The error is dominated by the energy spread of the beam in the storage ring. Using Eq. (1) and Eq. (2), the beam energy can now be calculated as:

$$E_1 = \left(\nu_{int} + \frac{f_{dep}}{f_o}\right)\cdot\frac{m_e c^2}{a} = 2899.8\,\mathrm{MeV}\,, \qquad (4)$$

where $\nu_{int} = 6$ is the integer part of the spin tune. This result is very close to the expected value of 2900 MeV. However, at this point the ambiguity between ν_{frac} and $1-\nu_{frac}$ could not be ruled out. Using Eq. (1) and Eq. (3), the second solution of the beam energy can be calculated as:

$$E_2 = \left(\nu_{int} + 1 - \frac{f_{dep}}{f_o}\right)\cdot\frac{m_e c^2}{a} = 2828.6\,\mathrm{MeV}\,. \qquad (5)$$

In order to distinguish between these two solutions, the beam energy was slightly increased and the measurement was repeated. This time the depolarization frequency was measured as:

$$f_{dep} = 1.0205 \pm 0.001\,\mathrm{MHz}\,.$$

This leads to the solutions:

$$E_1 = 2900.2\,\mathrm{MeV}\,, \qquad (6)$$

$$E_2 = 2828.2\,\mathrm{MeV}\,. \qquad (7)$$

Since the beam energy had been increased, the results in (4) and (6) must be the correct solutions.

REFERENCES

[1] S.C. Leemann, "Precise Energy Calibration Measurement at the SLS Storage Ring by Means of Resonant Spin Depolarization," Master Thesis, March 2002, ETHZ-IPP Internal Report 2002-02, http://simonleemann.ch/work.html.

[2] A.-S. Müller et al., "Energy Calibration of the ANKA Storage Ring," Proceedings of EPAC 2004, Lucerne, Switzerland.

[3] S.C. Leemann et al., "Precise Beam Energy Measurement at the SLS Storage Ring," Proceedings of EPAC 2002, Paris, France.

[4] P. Kuske et al., "High Precision Determination of the Energy at BESSY II," Proceedings of EPAC 2000, Vienna, Austria.

[5] C. Steier, J. Byrd, P. Kuske, "Energy Calibration of the Electron Beam of the ALS Using Resonant Depolarization," Proceedings of EPAC 2000, Vienna, Austria.

[6] L. Arnaudon et al., "Accurate Determination of the LEP Beam Energy by Resonant Depolarization," August 1994, CERN SL/94/71 (BI).

[7] Ya.S. Derbenev et al., "Accurate Calibration of the Beam Energy in a Storage Ring Based on Measurement of Spin Precession Frequency of Polarized Particles," Particles Accelerators 10 (1980) 177.

[8] J.M. Vogt et al., "Bunch Cleaning at the Canadian Light Source," Proceedings of PAC09, 2009, Vancouver, Canada.

STATUS OF THE FUTURE SPIRAL2 CONTROL SYSTEM

D. Touchard*, P. Gillette, C. Haquin, E. Lemaître, L. Philippe, E. Lécorché# (Ganil / Caen, France)
J.F. Denis, F. Gougnaud, J.F. Gournay‡, Y. Lussignol, P. Mattei (CEA-IRFU / Saclay, France)
P. Graehling, J. Hosselet, C. Maazouzi (CNRS-IPHC / Strasbourg, France)

Abstract

For the study of fundamental nuclear physics, the SPIRAL2 facility, a driver accelerator followed by a rare ion production process, will be coupled with the existing GANIL machine to provide light and heavy exotic nuclei at extremely high intensities. To ease the collaboration with several institutes on the control system design, EPICS has been chosen as the basic framework and a specific care has been taken concerning the software organization and management. While first operational interfaces for power supplies, faraday cups or beam slits are already operational, a triggered fast acquisition system for beam diagnostics, a radiofrequency control system, and an admittance measurement system are going to be achieved. First EDM supervision screens and high level tuning applications based on EPICS/XAL framework have been designed. The use of relational databases, on the one hand for the design of an environment to generate the EPICS databases, on the other hand to manage, set and archive meaningful values of the new facility, is under investigation. From the beginning of last year, two sources followed by their first beam line sections have been tested. Promising results are presented.

THE SPIRAL2 PROJECT

Overview

Following the recommendations of international committees and to fulfil the growing demand of the international physiscists community, in May 2005 the French Research Minister decided to build the new SPIRAL2 facility at GANIL laboratory (CNRS-CEA) in Caen (France) [1]. The project aims to enlarge multi-beam production using Isotope Separation On Line (ISOL) method. A superconducting LINear ACcelerator (LINAC) for light and heavy ions preceded by a radio frequency quadrupole (RFQ) will deliver up to 40MeV/A for 5mA deuteron, respectively 14.5MeV/A for 1mA heavy ion continuous wave (CW) beams [2]. These beams can be used for the production of intense Radioactive Ion Beams (RIB) involving the fusion, fission, transfer reaction mechanisms. More specifically, production of RIB with intense neutron-rich nuclei will be based on the fission of uranium targets bombarded either by neutrons produced by a first impact of the deuteron beam on a carbon converter, or by the direct deuteron or heavy ion beam impact. The RIB post-acceleration will be performed by the existing CIME cyclotron, which is perfectly suited to the separation and acceleration in the energy range up to 10MeV/A for the atomic masses between 100 and 150. SPIRAL2 beams after CIME can be reused in present experimental areas of GANIL (see fig 1).

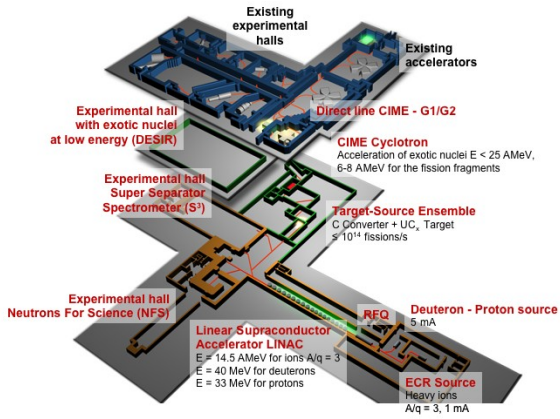


Figure 1: The SPIRAL2 and GANIL facilities.

Milestones

The first primary beams are expected in spring 2012 (phase 1) while the production process is planned more or less one year later (phase 2).

Some parts of the accelerator have been tested: the ion source and its low energy beam line have been in test at CNRS-LPSC (Grenoble) since 2008 and the deuteron source and its coupled beam line are progressively tested at CEA-IRFU (Saclay) since the beginning of this year[3].

Collaboration organisation

In order to build a large machine as SPIRAL2, an international collaborative effort has been made to establish the grounds for the design. A large team composed of people from CNRS, CEA, and international institutes is involved in the scheme. This is also the case as far as the command control system is concerned. The following three French laboratories, GANIL (Caen), IPHC (Strasbourg) and IRFU (Saclay) are currently designing and developing respectively the whole hardware and software command control system phase 1 architecture. For the phase 2, collaboration with the following three laboratories, LPSC (Grenoble), CENBG (Bordeaux), and LPC (Caen) is presently under consideration.

CONTROL SYSTEM

Main choices

The main choice of EPICS [4] as a common framework was early decided to ease pieces of software development and integration efficiency. A set of drivers already

*touchard@ganil.fr
#Spiral2 command control coordinator
‡Injector command control coordinator

developed inside this community has been picked to control the majority of equipment. 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[5].

Tuning applications read and write values via the EPICS Channel Access protocol. These values are hosted in EPICS IOCs accessing equipment directly with ADAS VME I/O cards, or Modbus/TCP field bus networks.

Software development

To ease software sharing a unique EPICS SPIRAL2 working environment, an equipment naming convention, and operational rules for interfaces have been specified. The use of SVN server, a versioning control software system, hosting specifics SPIRAL2 EPICS and JAVA directory skeletons, was decided to centralize all the pieces of software [6]. To tune a large facility as SPIRAL2, supervision screens developed with EPICS Extensible Display Manager (EDM), with the same template files and more sophisticated JAVA high level applications derived from the XAL [7],[8] framework, are under way. In order to investigate new EPICS innovative tools, an evaluation of the relatively new Control System Studio (CSS) integrated development environment has begun this year.

EQUIPMENT INTERFACES

Power supplies

To transport or control the beam along the facility, a set of 600 magnetic or high voltage pieces of equipment with specific power supplies driven through the MODBUS/TCP network protocol are needed [9]. Although different kind of power supplies have been already selected, for most of them a common mapping interface will be enforced between IOC and power supplies and a special care was taken about status feedback and tuning application interface. A first EPICS IOC database interface has already been developed and enhanced with an EPICS/GENSUB record.

Faraday cups

Faraday cup, a beam interceptive diagnostic [10], aims to measure through a VME/IOC beam intensity for either pulsed or continuous beam. Specific EPICS drivers were written for the fast acquisition boards ADAS ICV108/178 [11]. Last development improvements allow to set on the fly acquisition rate from 100K to 1.2M samples/s and dynamically select the piece of equipment to acquire. This system is being validated during beam tests.

Beam profilers

Profile monitors measure and visualize the beam dynamics. The EPICS IOC database and JAVA visualisation interfaces of secondary emission profiler have been recently developed. This development has particularly confirmed the necessity of the interface rules decided at the beginning of the project because of the special work induced by the mapping of complex data between IOCs and high level applications.

Emittance measurement system

Emittance measurement characterizes the horizontal and vertical transverse optical behaviour of the beam. The system is built over two scanner pods [12] that are controlled with brushless motors connected to an Oregon MaxV 4000S card. These pods require high voltage power supplies controlled with an ISEG VHQ202 M board or an ADAS ICV714 board. Even though an ADAS ICV150 board presently acquires values inside the faraday cups, the fast acquisition solution described above will be used for nominal performances. A state machine piece of software inside the IOC with a specific algorithm controls the scanner pods in order to build a whole emittance measurement.

Radiofrequency control system

Radiofrequency equipment such as choppers, bunchers, RFQ and LINAC cavities work at 88.0525MHz. They are independently powered by amplifiers controlled via the MODBUS/TCP protocol. IRFU laboratory has developed SPIRAL2 specific VME64x Low Level Radio Frequency (LLRF) boards which integrate, Field Programmable Gate Array (FPGA) to regulate the amplitude-phase of each cavity. An EPICS device/driver has been developed for this card, along with a new type of record. This record also implements an acquisition mode needed for commissioning any piece of software, electronic development or cavities [13]. The prototype of the VME board and RF card and the principle of the Proportional-Integral-Derivative (PID) digital control of phase and amplitude have been validated.

DATABASE ORGANISATION

Equipment characterization

Considering 4000 new pieces of equipment expected to control the facility which will be mainly managed by non EPICS users, led the team to consider a specific equipment INGRES database associated to a friendly JAVA user interface. Each type of equipment should be driven by generic EPICS development which consists of database files with macro substitution. A specific IOC generator will produce start-up IOC files filled with each equipment characterization. This organization was successfully tested with a power supply generic development.

Beam parameters

The SPIRAL2 facility will produce and accelerate a range of beams including deuterons and heavy ions, with different optics, and trajectories. In order to tune this facility efficiently, a beam management process as the

one already existing at GANIL is now foreseen [14]. A first application based on the XAL framework described below and based on an Ingres database has been successfully tested this year.

Archives

The first evaluation of the EPICS CSS archiving system based on a MySQL 5.5.1 database has shown a need of disk space of about 10To to store continuous activity 7d/7, 24h/24 during 8 weeks run for 400 significant machine parameter values having 10Hz change rate. To keep performances compatible with the CSS data browser, a study on MySQL is underway for partitioning tables and tuning servers.

HIGH LEVEL APPLICATIONS

Simulations

Very extensive calculations using TOUTATIS and TRACEWIN [15] codes developed by the IRFU laboratory have been performed to simulate beam behaviour and losses, especially in RFQ and LINAC section. These codes could generate equipment theoretical values that could be inserted in the beam parameter database described above. The software gateway between TRACEWIN and the parameters database has been validated last year.

Common frameworks

The whole XAL software could not be reused as is, due to SPIRAL2 specificities such as multi ion species accelerated. The evaluation of this framework has shown that it could increase significantly development efficiency. First applications developed with this graphical interface or reusing some code or pieces of software, such as accelerator configuration, profiler display, beam adaptation or optimisation, are underway.

BEAM TESTS

Beam line section LEBT1 tests performed at LPSC as shown in fig. 2 were an important step to validate the associated control command.

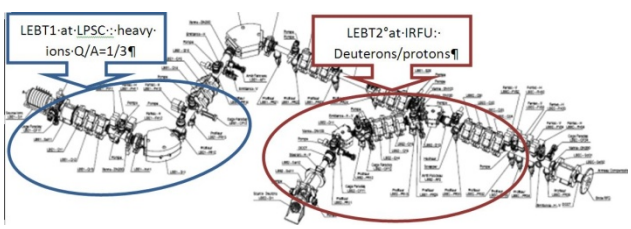


Figure 2: LPSC and IRFU beam tests.

A Labview PC controlling the heavy ion source and a set of EPICS Linux/PCs and VxWorks/VME crates controlling the line equipment communicate through a Labview / EPICS gateway supplied by National Instruments. For EPICS front end part, power supplies and Faraday cups interfaces, the fast data acquisition system, dialogue with the PLCs, emittance measurements have been integrated. Tuning applications such as the correlation of all parameters and analysis of the heavy ion source have been validated. In order to study the beam characteristics, a special use of TRACEWIN connected to a Faraday cup and a legacy profiler diagnostics, and controlling power supplies has successfully optimized the beam transmitted. In the same state of mind, and in order to control the deuteron source with a fully EPICS control system, the beam line section LEBT2 is being tested at IRFU.

ACKNOWLEDGMENTS

As the control system is the core interface with many pieces of equipment, special thanks for the contribution of many people at IRFU, IPHC, LPSC and GANIL laboratories, without whom the SPIRAL2 control command could come out.

REFERENCES

[1] S. Gales “SPIRAL2 at GANIL: A world leading ISOL facility at the dawn of the next decade”, International Workshop on Nuclear Physics 28th Course 2006
[2] T. Junquera, P. Bertrand, R. Ferdiand, M. Jacquemet, “The high intensity superconducting LINAC for the SPIRAL2 project at GANIL”, LINAC2006, Knoxville, USA
[3] T. Junquera, “Status of the construction of the SPIRAL2 accelerator at GANIL”, LINAC08, Victoria BC, Canada
[4] Matthias Clausen, “EPICS FUTURE PLANS”, ICALEPCS07, Knoxville, Tennessee, USA
[5] E. Lécorche, “Preliminary implementations for the new SPIRAL2 project control system”, PCaPAC08, Ljubljana, Slovenia
[6] D. Touchard, “The SPIRAL2 command control software organization and management”, ICALEPCS 2009, Kobe, Japan
[7] J. Galambos, “XAL application programming framework”, ICALEPCS2003, Gyeongju, Korea
[8] T. Pelaia, “XAL status”, Proceedings of ICALEPCS07, Knoxville, Tennessee, USA
[9] D. Touchard, “A MODBUS/TCP-BASED power supply interface”, PCaPAC08, Ljubljana, Slovenia
[10] C. Jamet, “Injector diagnostics overview of SPIRAL2 accelerator”, Proceedings of DIPAC 2007, Venice, Italy
[11] F.Gougnaud, “The first steps of the beam intensity measurement of the SPIRAL2 injector”, ICALEPCS2009, Kobe, Japan
[12] P. Ausset, “SPIRAL2 injector diagnostics”, DIPAC09, Basel, Switzerland
[13] M. Di Giacomo, “Status of the RF system for the SPIRAL2 LNAC at the beginning of the construction phase”, LINAC 2006, Knoxville, USA
[14] E. Lécorché, “Use of the INGRES RDBMS inside the new GANIL LINUX based control system”, ICALEPCS2005, Geneva, Switzerland
[15] Logiciels at IRFU/SACM, http://irfu.cea.fr/Sacm

SETTINGS MANAGEMENT WITHIN THE FAIR CONTROL SYSTEM BASED ON THE CERN LSA FRAMEWORK

J. Fitzek, R. Mueller, D. Ondreka, GSI, Darmstadt, Germany

Abstract

A control system for operating the future FAIR (Facility for Antiproton and Ion Research) accelerator complex is being developed at GSI. One of its core components is the settings management system.

At CERN, settings management and data supply for large parts of the CERN accelerator complex is done using the LSA (LHC Software Architecture) framework. Several concepts of the LSA framework already fit the FAIR requirements: Generic structures for keeping accelerator data; modular design; separation between data model, business logic and applications; standardized interfaces for implementing the physical machine model. An LSA test installation was set up at GSI and first tests were performed controlling the existing GSI synchrotron SIS18 already with the new system.

These successes notwithstanding, there are issues resulting from conceptual differences between CERN and FAIR operations. CERN and GSI have established a collaboration to make LSA fit for both institutes, thereby developing LSA into a generic framework for accelerator settings management. While focussing on the enhancements that are necessary for FAIR, this paper also presents key concepts of the LSA system.

FAIR

The international FAIR facility with its nine new accelerator installations will be built at GSI, using the existing linac and synchrotron SIS18 as injectors (see Fig. 1).

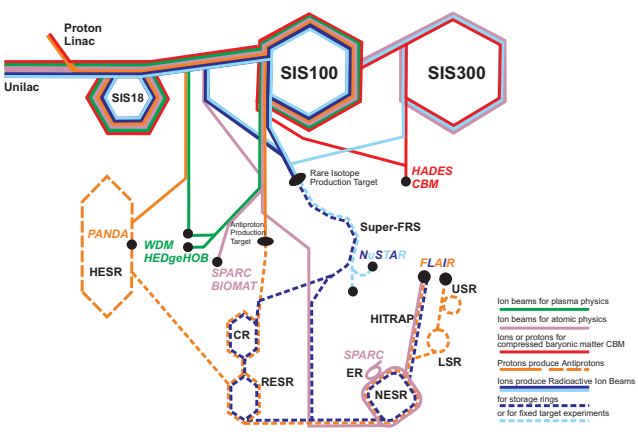


Figure 1: GSI/FAIR beamlines, P. Schuett, GSI 2010.

Central aspect is an increased number of research programs resulting in up to five beams in parallel with pulse-to-pulse switching between different particle types. The future facility will be controlled by a new control system which will be able to support all aspects of the complex GSI/FAIR operations on a common technical basis [4]. The future control system is designed at the moment, keeping well working and proven principles while adopting new methodologies where beneficial.

Important aspects of the control system are generation of settings and data supply. It was evaluated and decided to use the existing LSA framework from CERN for settings management and data supply within the FAIR control system. A collaboration with CERN was set up with joint development effort put into future LSA development [1].

LSA - THE LHC SOFTWARE ARCHITECTURE

LSA was developed at CERN starting in 2001 and is now the core controls software component for settings management and data supply within the CERN control system. For a detailed description of LSA see [2].

LSA - Functional Overview

The LSA system was designed in a generic way and provides clear separation between data model, business logic and applications. Its modular structure allows institute specific implementation to be easily plugged in.

The system covers all important settings management aspects: optics (twiss, machine layout), parameter space, settings generation and management, settings modification (trim), propagation from physics to hardware parameters, operational and hardware exploitation (equipment control, measurements), and beam based measurements.

An accelerator within LSA is modeled by defining its parameter hierarchy – from top level physics down to hardware parameters. Using the optics, the LSA system can already calculate good initial settings. Corrections can be applied to any level of the hierarchy, resulting in a consistent change of many devices at the same time. As an example for a part of such a hierarchy at GSI, see Fig. 3. The LSA system consists of different functional building blocks, which among other benefits entitle physicists to implement the machine model themselves in a structured and simple way.

LSA - Technology Stack

LSA is written in Java and uses the Spring framework, that provides a light-weight container for the Java platform, dependency injection, aspect oriented programming (AOP), testing framework, remoting and transactions. An overview of the LSA software stack is shown in Fig. 2.

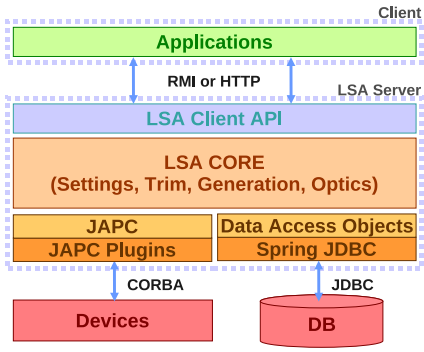


Figure 2: LSA software stack, G. Kruk, ICALEPCS 2007.

LSA is visible to the applications trough a series of façade classes that group the functionality of LSA by topics (setting management, hardware access etc.). They represent a stable and backward compatible interface which separates applications from business logic, thus applications can concentrate on presenting information.

Communication with the devices is done through a powerful abstraction layer called JAPC (Java API for Parameter Control) [3], that hides middleware specifics and thus allows access to all devices through the same interface.

LSA AT GSI

The collaboration on LSA started in 2007 with two software developers from GSI working for 1.5 years on site in the LSA team during the LHC commissioning and startup. Since then the collaboration is well established. In 2008, an LSA test system was set up at GSI.

Setting up an LSA test system

The LSA system runs out-of-the-box given an empty LSA database with just a few tables prefilled. First steps include setting up an Oracle database instance and an LSA test server, which is a standalone Java process. For the small number of missing software references (e.g. to CERN's online model server or role based access system), a dummy implementation needs to be provided which fulfills the interface. Since the LSA system is data-driven, the next step is to import the accelerator layout into the database, such as static information about accelerators, beamlines and devices. As a result of this initial setup, generic LSA applications deployed via Java WebStart are already running.

After a new JAPC plug-in for the existing GSI middleware was developed, first calls to devices proved that the environment was correctly set up.

Implementing GSI accelerators within LSA

A project team consisting of machine physicists and software developers from different groups at GSI started modeling the existing synchrotron SIS18 within the institute specific part of LSA: defining the parameter hierarchy, implementing propagation rules, importing optics and defining test cycles. Since the implemented rules were written in a generic way, even test cycles for the future FAIR synchrotron SIS100 have already been successfully generated. Next step will be to look at the existing GSI storage ring and its representation within the new system.

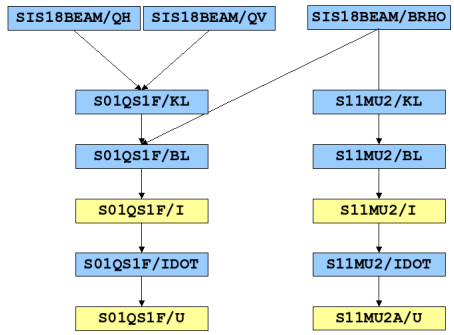


Figure 3: Example of an LSA parameter hierarchy at GSI.

While modeling the existing accelerators in LSA, the project team mainly focusses on the LSA concepts and the question, whether they are really generic enough to fit all needs of complex and parallel accelerator operations.

First test with beam

Keeping in mind the challenge of putting the new control system for FAIR in place while the existing facility is running, it is however vital, that the new system will be commissioned with parts of the existing machine.

For the settings management and data supply part with LSA, this includes testing the new system already with the existing accelerators. As a first milestone, a successful test with beam was performed in march 2010 with the existing synchrotron SIS18. Several scenarios were tested: operations with one and two cavities and one and two shot extraction using a cycle with fast extraction.

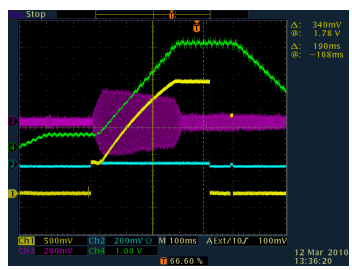


Figure 4: SIS18 test: single shot extraction.

Extending LSA

The flexibility of accelerator operations at GSI put a new view on the LSA system. In particular while modeling the SIS18, certain restrictions were found within the LSA machine model, resulting from the rather static operation of the accelerators at CERN. Based on these observations, requirements were collected. For some of these topics a final solution has been agreed upon and implementation has already started.

One new feature within LSA which will be implemented within the collaboration is the flexibility of cycle length: length of cycles and therefore length of specific functions like the dipole current can vary due to applied trims. This feature will be heavily used at GSI, where e.g. extraction energy is frequently trimmed and the corresponding adaptation of the cycle length is indispensable for the optimization of the duty cycle. This of course presumes a flexible timing system with no predefined base cycle length.

Another feature will be the support for modeling a full chain of accelerators, especially modeling inter-accelerator dependencies, which is necessary for FAIR. Also at CERN, the focus shifts towards controlling the full accelerator chain. This change in perspective is related to the fact, that now the same control system is used for many accelerators at CERN. The idea is e.g. to connect the extraction energy of one accelerator with the injection energy of the next accelerator in the chain and automatically trim settings for the whole affected chain when changed in one place. First brainstorming on this topic has started and will be continued in the near future.

So far, it seems that all of the requirements now coming from GSI are also of interest for CERN and that their implementation will be part of the LSA core system. The goal of both involved parties is clearly to make LSA as generic and flexible as necessary to be able to really fulfill all requirements, that arise from complex accelerator operations.

Technically implementing new features in the LSA core system is encouraged by the use of the Spring framework. It easily allows plugging in test implementations by one party using XML configuration while the existing implementation remains untouched. This also supports using one repository for the LSA core system even while realizing new features.

However, also institute specific implementation like access to devices, accelerator specific physics propagation rules etc. fit into the LSA concept: they reside in institute specific modules which complement the core functionality by implementing the respective interfaces. Even though it is planned to manage those specific software modules locally at the institute site in the future, at the moment the GSI modules still reside at the CERN repository, benefitting from CERNs build and release environment.

Development of LSA based applications

In addition to the existing generic LSA applications there is the need for new applications developed at GSI which fit the operators workflow. Since the LSA business logic is well separated from the applications and encapsulated by GSI specific façade classes (where only a subset of full LSA features is made visible to application developers), applications can focus on displaying information. Additionally standard prefilled GUI elements and a stable API substantially ease application development and also encourage others to write applications based on LSA.

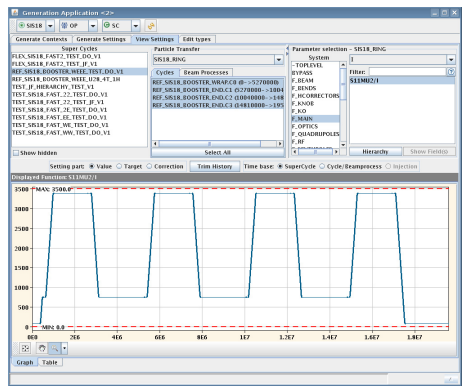


Figure 5: LSA application showing a SIS18 supercycle.

SUMMARY/OUTLOOK

The prototype installation and successful first tests with beam proved that the LSA framework already fits the requirements for settings management and data supply for single accelerators within the FAIR control system. From a technical perspective it was easy to install and to set up the system in its initial state. The biggest effort was to implement the accelerator model using the LSA framework.

New requirements arise from the flexible GSI/FAIR accelerator operations and from the necessity to model the whole accelerator complex within LSA. The corresponding enhancements of the LSA framework are implemented within the collaboration. In this way LSA evolves into a generic and flexible settings management framework for complex accelerator facilities.

REFERENCES

[1] R. Mueller, J. Fitzek, D.Ondreka, “Evaluating the LHC Software Architecture for data suppy and settings management within the FAIR control system”, ICALEPCS’09, Kobe, Japan, THP012.

[2] Grzegorz Kruk et al, “LHC Software Architecture (LSA) - Evolution Toward LHC Beam Commissioning”, ICALEPCS’07, Knoxville, Tennessee, USA, WOPA03

[3] V. Baggiolini et al, “JAPC - the Java API for LHC Timing Parameter Control”, ICALEPCS’05, Geneva, Switzerland, TH1.5-8O.

[4] Ralph C. Bär et al., “Development of a New Control System for the FAIR Accelerator Complex at GSI”, ICALEPCS’09, Kobe, Japan, TUP107.

INTEGRATION OF PROGRAMMABLE LOGIC CONTROLLERS INTO THE FAIR CONTROL SYSTEM USING FESA

R. Haseitl, C. Andre, H. Bräuning, T. Hoffmann, R. Lonsing, GSI, Darmstadt, Germany

Abstract

For the upcoming 'Facility for Antiproton and Ion Research' (FAIR) at GSI, the Front End Software Architecture (FESA) framework built by CERN has been chosen to serve as front-end level of the future FAIR control system [1]. All beam diagnostic devices of FAIR will be controlled by FESA classes that are addressable by the new control system. The connectivity to the old control system is retained, since both control systems will be in operation contemporaneously for several years. Commercially available Programmable Logic Controllers (PLCs) have been installed as part of Beam Induced Fluorescence (BIF) monitors to replace outdated network attached devices and to improve the reliability of the BIF systems. The new PLC devices are controlled by FESA classes which are addressed from the existing C++ software via Remote Data Access (RDA) calls. This contribution describes the system setup and the involved software components to access the PLC hardware.

THE BIF SYSTEM

Beam Induced Fluorescence monitors determine the transverse beam profiles with minimum beam disturbance [2]. The measurement principle is based on the excitation of gas molecules by the passing ion beam in the beam pipe. The emitted photons are measured by digital CCD cameras with image intensifiers to ensure single photon detection. Using two cameras installed above and sideways of the beam pipe, the horizontal and vertical beam profiles are measured simultaneously. Currently, there are four BIF monitors installed in GSI accelerators and transfer lines. For the next years, a final number of seven monitors is anticipated.

Hardware

Each camera has a remote controllable iris to adjust the light intensity illuminating the image intensifier. A smaller aperture of the iris also increases the depth of field. This results in a larger properly focused area in the obtained image. The amplification of the image intensifiers can be adjusted by setting two voltages for the different amplification stages.

The aperture of each iris as well as the amplification of the image intensifiers has formerly been controlled with a self-built, Ethernet connected module, containing several digital-to-analogue converters (DACs). During long term runs of the system, these modules crashed non-deterministically after some hours or days of operation. For the FAIR project, a more reliable solution was desired. The setting of voltages is a common task for PLCs, so this commercially available and field-tested solution was selected.

Software

The software controlling all BIF devices, including irises and image intensifiers, is called ProfileView [3]. The communication with the old hardware is performed via a standard TCP connection. New settings are sent to the device, which replies with an acknowledge message. The communication channel is kept open continuously, to detect failures as soon as possible.

After extensive testing of the system, it was decided to replace the faulty devices by PLCs. To control the new hardware, the ProfileView software was adapted to support both hardware variants.

Figure 1: One of the 'satellites' of the BIF installation. It controls two BIF monitors and features two sets of control devices (from left to right): Power Supply, ET 200M controller, SM322 relay element with eight outlets, two SM332 12-bit DACs.

PLC HARDWARE

As PLC hardware, the SIMATIC system from Siemens [4] was selected (see Fig. 1). The PLC is able to set voltages for each iris, the intensifiers and calibration LEDs which had to be controlled manually before. Furthermore, an easy to use remote reset capability for supplemental hardware devices of the BIF system is realized by relay modules. The relays support switching of 230 volt supply voltage. Devices of the BIF system like the cameras or the gas flow control can be restarted remotely in case of errors.

The hardware was installed at different locations along the linear accelerator. A schematic of the system is depicted in Fig. 2. The main controller and the Ethernet communication module are located in an electronics room. The distributed sub-systems with local control units, relays and DAC devices, so-called 'satellites', are located near the BIF hardware in a radiation safe area.

The system consists of the following Siemens SIMATIC components:

- S7-300 - the main controller
- CP343-1 Lean - for Ethernet communication
- ET 200M - the satellite controller
- SM322 - relay with eight outlets
- SM332 - 12-bit DAC with four outlets

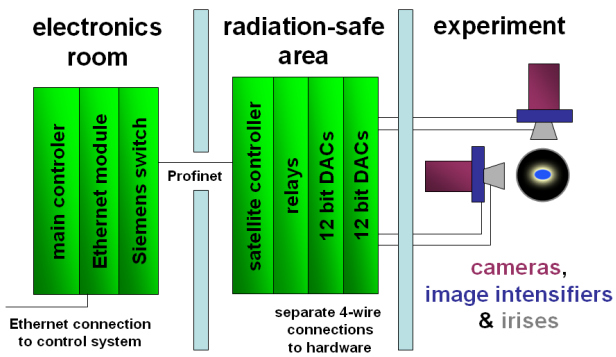


Figure 2: PLC installation.

Communication from the main controller to the satellites takes place via Profinet.

To ensure correct voltages at the BIF hardware, the connections from the DACs to the devices are made in 4-wire technique, to sense the voltage loss over the cable length. In this way, conduction losses are compensated and the applied voltage at the hardware matches the desired voltage in the software.

FESA CLASS AND PLC SOFTWARE

The FESA framework developed by CERN [5] will be the front-end level of the FAIR control system. A FESA class is typically developed by the hardware specialist of a device and provides read/write access on the device's registers. FESA runs on PowerPC or Intel based VME CPUs and on standard Linux PCs. The connection from the control system to the FESA class is established via RDA calls defined in the CERN Middleware (CMW) library. A FESA class allows incoming connections from multiple applications. If an application subscribes to data changes in the device, the FESA class will notify the application in case of new values.

FESA classes

To access the PLC via the control system, two FESA classes are in operation: One very simple class ('BIFPLC') for basic communication with the PLC and a more complex class to perform calculations and monitor the PLC status ('BIFControl'). The BIFControl class uses the communication functionality via an FESA equipment link. It performs the transformation of values from the graphical user interface (GUI) into the bitwise register representation needed for the DACs in the PLC. Furthermore, the instantiation of BIFControl defines failsafe values for each DAC. To protect the image intensifier system, all voltages are limited within BIFControl between defined thresholds. The relay contacts for 230 volt switching are controlled by BIFControl, too.

Communication with the PLC

The BIFControl class has full read/write access to the memory of the PLC controller via the network. Write cycles are usually initiated whenever the class gets new settings from the GUI. Read cycles can be initiated by the GUI or periodically by the class itself.

From the PLC controller's point of view, the data transfer consists of reading from or writing to the same memory space. This is typically done from within the periodically executed organization block 'OB1'. This memory access is completely asynchronous to the access by the BIFControl class. To facilitate the access to complex data structures from within the PLC, scripts and a web interface exist [6]. These tools analyse the design of the BIFPLC class and create short function blocks to access the data structures in the PLC's memory. The scripts are able to generate Step 7 code for the Siemens PLC used at GSI as well as code for Schneider PLCs.

SOFTWARE INTEGRATION

The ProfileView software for BIF monitors is entirely written in C++. To access the BIFControl class for PLC control, a directory server is asked via CMW for a device handle. The returned handle offers access to get, set and subscribe methods. CMW exists as a static library and is linked into the ProfileView executable. It contains the RDA functions and data access methods to connect to FESA classes and to extract user data from RDA telegrams. RDA itself is based on CORBA.

It is possible to set single values as well as multiple data fields at once. The BIFControl class can be accessed from different applications or expert programs at the same time. Therefore, a notification on any value change is

desirable. For this purpose a subscription handler is installed, listening to value changes of BIFControl. By invoking the monitorOn() function on the device handle, a user defined callback function can be declared. Every change of any value in the class leads to a call of the specified callback function, containing the new values. This way, ProfileView gets notified, if PLC values are changed by any other application.

The subscription handler is programmed as a separate thread to avoid blocking the application when it is waiting for new data. Once an updated data set arrives at the handler, the new data is processed inside ProfileView and shown in the GUI (see Fig. 3).

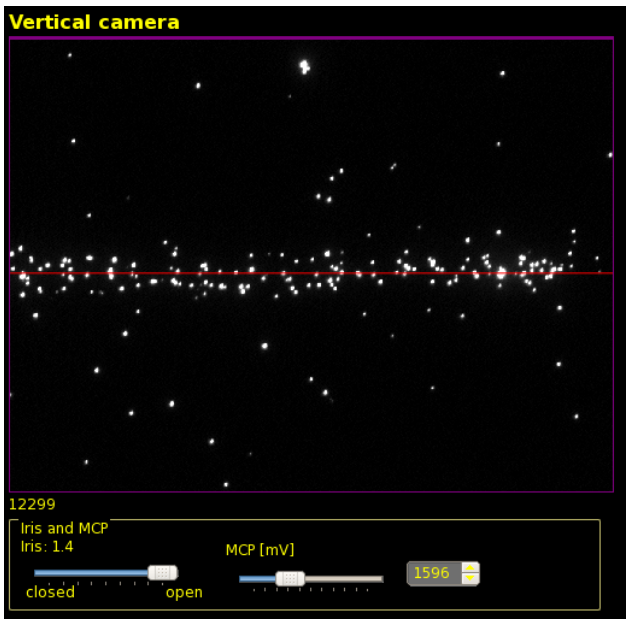


Figure 3: Original BIF image with GUI elements for iris and image intensifier control.

The values of the image intensifier amplification are directly passed to the GUI and shown in millivolt units. Iris values are treated specially: since there are small differences between the remote controllable iris devices, each iris has to be calibrated separately before it is built into a BIF monitor. The calibration provides a millivolt value for each aperture of a particular iris. These settings are stored in the initialization file of the BIF monitor. If a new iris value is set, ProfileView sends the appropriate millivolt value to the FESA class, which in turn sends the bitwise representation to the PLC. The chain is executed in reverse when the FESA class pushes a new iris value to subscribed client applications.

The relays are implemented as a bit array inside ProfileView and the FESA class. If a relay should be toggled, the others have to keep their state. If the user wants to set a new relay value, it is matched with the stored state of the bit array and sent to the hardware.

The integration of the PLC functionality into ProfileView hides the hardware specific part from the user. The GUI looks the same, no matter if the old hardware or a PLC is used for iris and image intensifier control. The relays are only available when a PLC is used.

CONCLUSION AND OUTLOOK

Since there are several applications for PLC controllers planned for the future FAIR facility, this prototype is a good test of the system reliability and the integration into the existing control system.

The PLC hardware turned out to be very stable and reliable in operation. Using the 4-wire technique, the voltages at the hardware are accurate to ± 0,01 volt. During commissioning and several weeks of tests, the PLC did not crash and never lost its connection to the FESA class.

Currently (2010) one BIF monitor is equipped with a PLC system and one more is prepared for the hardware exchange. During 2011, the existing and any additional BIF monitors will be updated with PLCs. One PLC system will be installed in a radiation exposed area in the transfer line from the linear accelerator to the synchrotron. The dose will be measured directly at the PLC to gather information about the system's radiation hardness.

In 2011 a new version of FESA (3.0) will be finalized by CERN in collaboration with GSI. It will simplify the development of PLC FESA classes. Instead of having two FESA classes for PLC control, the complete PLC functionality will be a static library which can be linked to any FESA class.

REFERENCES

[1] T. Hoffmann, M. Schwickert et al., FESA at FAIR - The Front-End Software Architecture, PAC'09, Vancouver, Canada, FR5REP009, (2009).

[2] F. Becker, P. Forck et al., "Beam Induced Fluorescence Monitor for Transverse Profile Determination", DIPAC'07, Venice, (2007).

[3] R. Haseitl et al., "ProfileView – A Data Acquisition System for Beam Induced Fluorescence Monitors", DIPAC'09, Basel, May 2009, MOPD42, p.134 (2009).

[4] Company Siemens, www.siemens.com

[5] M. Arruat et al., "Front-End Software Architecture", ICALEPCS'07, Knoxville, TN, (2007).

[6] Frank Locci, CERN, BE/CO-FE, Frank.Locci@cern.ch

FESA BASED DATA ACQUISITION FOR BEAM DIAGNOSTICS AT GSI

T. Hoffmann, H. Bräuning, R. Haseitl, GSI, Darmstadt, Germany

Abstract

In view of the upcoming Facility for Antiproton and Ion Research (FAIR) at GSI with its increased complexity in beam control and diagnostics, the decision was taken to use the well-tested CERN made Front-End Software Architecture (FESA) as the lowest level of the new control system [1,2]. In the past years, the current stable FESA framework (Version 2.10) has been adapted and installed at GSI, with the major part of the adaptation being the different machine timing models of GSI and CERN. With a stable environment at hand, all current and new beam diagnostic related data acquisition systems will be implemented with FESA. To demonstrate the suitability of FESA for demanding data acquisition problems with high data rates or large amounts of data, two different projects such as the Tune Orbit and POSition measurement (TOPOS) and the Large Analog Signal Scaling Information Environment (LASSIE) are presented. Experiences with implementing standard interfaces such as CAN, GigE and PLCs in FESA applications as well as a move towards low cost Intel based controllers like the Men A20 VME controller or industry PCs running a real time Linux will be discussed.

THE FESA ENVIRONMENT AT GSI

Besides the development of the next generation FESA 3.0 environment by staff of the GSI controls department in collaboration with CERN CO/FE, the GSI beam diagnostic department (BD), which is responsible for the layout of the FAIR beam diagnostics DAQ system, is developing FESA 2.10 classes for dedicated BD systems. These efforts are made to show the feasibility of all expected data acquisition requirements and to train programming of the front-end part of the new control system for FAIR. At the beginning of 2010 the FESA 2.10 installation and integration at GSI was fully accomplished. The environment resides on a powerful blade system, which is the new mainframe of the GSI control system providing NFS based access to all front-end controllers (FEC) and to all branches of code development. Basic information on FESA is given in [1,2]. The main parts of the FESA systems are:

Operating System

At present the operating system (OS) of the GSI control system is a Red Hat Enterprise Linux Server release 5.5 with kernel 2.6.18-92 - x86_64. The OS for the FECs is Scientific Linux CERN 5.4 with kernel 2.6.24.7-rt27, which contains patches for real-time support.

Supported FEC Hardware

FESA 2.10 provides cross compilers for Intel and PowerPC based CPUs. For maintainability reasons the following FEC systems are supported by GSI:

- Standard Industry PC
- Kontron KISS PCI760 with PXEBoot, diskless, Intel AMT remote management system
- MEN A 20 VME CPU with PXEBoot, diskless.

For applications requiring real time behaviour the CES RIO3 CPUs with Lynx OS can be used as an exception.

For the time being the upcoming xTCA for Physics standard [3] as a new form factor is under evaluation for the usage at FAIR. For the tests an Adlink AMC-1000 CPU in an ELMA xTCA-6 frame were chosen. After integration of the diskless system into the control system, the installation will be tested with high bandwidth applications such as GigE video imaging and analog data sampling.

Timing

FESA 2.10 is strongly dependent on the CERN timing system and its timing receiver hardware, which is different to the existing GSI timing. To gain efficient use of the FESA RT action feature a dedicated FPGA based GSI-CERN timing converter was developed. It allows to use the CERN timing receiver hardware with the GSI timing. Although some purely CERN specific features are not available, this converter allows to trigger RT actions by accelerator timing events in a multiplexed beam operation for all three GSI accelerators (UNILAC, SIS, ESR).

JAVA Graphical user interface

To provide GUIs for the developer as well as for the users such as machine operators or system experts the JAVA based concept of CERN was chosen. It consists of the Java API for Parameter Control (JAPC, [4]) and CERN libraries such as the JDataViewer and the CERN middleware (cmw-rda). Due to the JAVA web-start functionality and the general JAVA platform independence, the GUI may be used at any office at GSI.

TUNE, ORBIT, AND POSITION MEASUREMENT (TOPOS)

The first test project for FESA and its related middleware and GUI solutions at GSI was a development for the tune, orbit and position measurement (TOPOS) at the heavy ion synchrotron SIS in collaboration with Cosylab and Instrumentation Technology, Slovenia. The development of the modular extendible TOPOS was performed also in preparation for the FAIR project and the usage at the FAIR synchrotrons. The data acquisition concept is well described in [5].

This very demanding system, with data rates up to

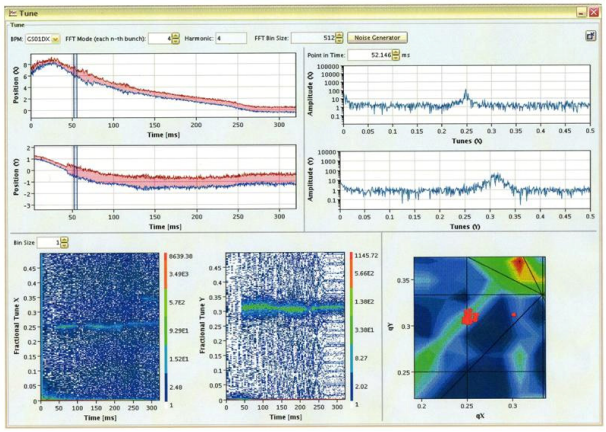


Figure 1: The FESA based TOPOS system showing horizontal and vertical tune measurements on excitation of the beam at $2*10^8$ U^{73+} ions per bunch.

700MByte/s, showed good results with respect to performance, stability, and usability. After benchmarking tests with the former beam position monitoring (BPM) system it will be raised to operational status at GSI. Fig. 1 gives an impression of the online tune measurement.

LARGE ANALOG SIGNAL SCALING INFORMATION ENVIRONMENT (LASSIE)

LASSIE is the new FESA based DAQ project to distribute and analyze a large quantity of beam diagnostic related analog and digital signals. It consist of FESA based data acquisition classes and JAVA GUIs. Recently the readout of a scaler array with 192 channels for SIS and connected beam line data was implemented in FESA. It is based on a VME system with six SIS3820 Multiscalers [6] and a dedicated timing receiver board. Scaler input consists of signals from beam loss monitors, experiment counters and other data like accelerator rf, current transformers etc. via a voltage-to-frequency converter. The scalers can be latched with a frequency of up to 1 MHz which provides fine-grained information about the spill structure. The GUI framework provides general GUI and non-GUI components like for example data structures, settings manager and a help system for rapid application development. Current applications include integrated counter values between selected machine events, spill structure analysis and trending.

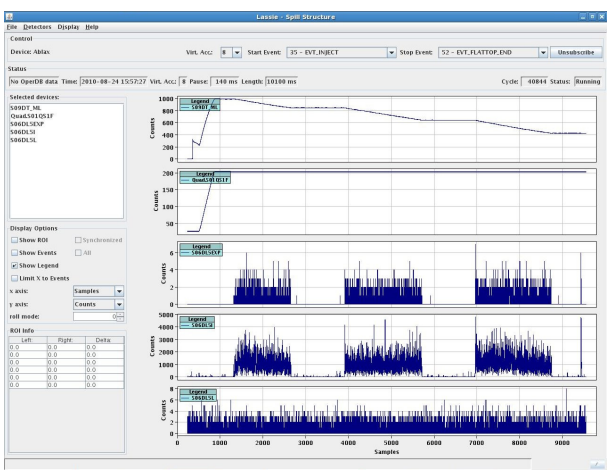


Figure 2: LASSIE: FESA based spill structure analysis of synchrotron signals (from top to bottom: current transformer, quadrupole ramp and beam loss monitors).

The system is now accessible from the accelerator control room for testing and will replace the Kylix based ABLASS [7] system with all functionalities. At typical scaler latching frequencies for normal operation around 100 to 1000 Hz, the FESA class can easily handle the readout of all 192 scaler channels. Using two memory banks, the FESA class allows GUIs to access the data of the just completed spill while acquiring the data for the current spill. In order to reduce the network load, the GUI applications use the filter mechanism of FESA to request for example the spill structure of only those channels which are displayed.

Another bottleneck for high latching frequencies is the transfer of data via the VME backplane. According to the SIS3820 specifications [6], the transfer rate via the VME backplane is limited to about 50 MByte/s for MBLT64 block transfer. To compare the rate capabilities of the older PowerPC based CES RIO3 CPUs and the new Intel based Men A20 CPUs, a test system with a single 32 channel scaler and a virtual machine cycle of 2000ms runtime and 150ms pause was set-up. Figure 3 shows the maximum number of scaler channels which could be read out by the FESA class without any connected GUI clients as a function of the latching frequency. Scaler access was done via block transfer (BMA) on the RIO3 and DMA on the Men A20. The Men A20 board consistently has a higher data throughput with a maximum measured rate of 34 MBytes/s. A total CPU load of only 10% indicates that this rate is limited by the transfer from the scaler module to the memory and not by data handling in the FESA class. In contrast, a maximum data rate of 22 MBytes/s

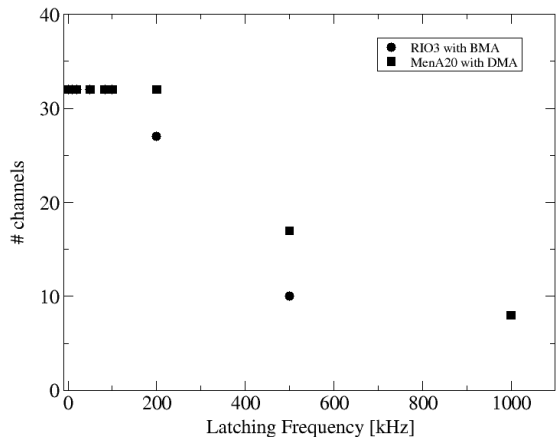


Figure 3: Number of scaler channels which can be readout as a function of the latching frequency.

was measured with the RIO3. This is accompanied by a strong increase in the CPU load, which indicates that the processor power is the limiting factor. However, it must be noted that this superiority of the Men A20 is only valid in the case of DMA usage. For direct access of a single register, the RIO3 has a slight advantage with a measured data rate of 1.4 MBytes/s as compared to 1.2 MBytes/s for the Men A20 board. In addition, setting up the DMA transfer takes some time and is thus suitable only for the transfer of large blocks of data via the VME bus.

Moreover, the Men A20 CPU with its 1000 Mbit/s Ethernet interface also allows a higher data transfer rate to the GUI application as compared to the RIO3. Thus the RIO3 CPU has been phased out in beam diagnostics applications and is replaced with the Men A20 CPU.

Future applications of the LASSIE system may include direct readout of ADCs for pulse-height analysis or TDCs for even more detailed spill structure analysis. For FAIR it is estimated that more than 1000 channels, distributed over the complete campus, will have to be read out. Preparatively the current system will be used as a test setup for a distributed DAQ system for proper intra-cycle data correlation.

SCADA APPLICATIONS

In addition to sophisticated DAQ systems, beam diagnostic devices depend strongly on technical subsystems such as pressurized air actuators, stepper motors, gas flow meters, high voltage power supplies and remote control operations. Such control requirements are also to be handled with FESA as the connector between the device and the GUI.

For devices like actuators, flow meters and such, a field-bus system will be established. For the time being the Programmable Logic Controller (PLC) Simatic S7-300 from Siemens is under evaluation. A description of the system and the connected beam induced fluorescence (BIF) measurement is given in [8]. Essential for this PLC setup was the interfacing with FESA, which was achieved using the IEPLC tool [9] from CERN. It creates Simatic code for data block exchange via Ethernet, which exchanges data with a predefined FESA 2.10 PLC class. The handling of the BIF system, e.g. control power for micro channel plates and camera iris regulation by use of FESA is now operational. The required calls to get and set data from the FESA class are implemented in a QT [10] based GUI.

The integration of the multi-channel high-voltage power supplies like the CAEN SY1527 into the control system is a must. A FESA class was developed which accesses the SY1527 system via Ethernet connection. The FESA class provides access to all channels at once for e.g. shutdown procedures, but also single channel access from application GUIs, where only a subset has to be controlled. This is achieved by extensive use of the filter mechanism provided by FESA for its properties. Safety is easily enhanced by the FESA class included monitoring, logging and alarm options.

A strict demand for all FAIR DAQ systems is the remote control access to all crates and systems, preferably via Ethernet. In some cases hardware has to be used, which allows only CAN bus access, for which a FESA class was developed.

OUTLOOK

In preparation for the FAIR project and the realization of the beam diagnostic DAQ system, all significant BD requirements, such as readout of high data rates, video imaging, distributed systems, slow controls, etc. were realized with FESA. The results are very satisfying and give confidence for the usage of FESA at FAIR. All new BD-DAQ systems for the existing accelerators will be realized with FESA to train developers and operators on the new technology. The new FESA Version 3.0 is expected to be released soon. By separating the FESA framework into a general and a lab-specific part, the new version will accommodate the GSI environment more suitably. As soon as a production quality will be reached, the current BD FESA classes will be ported to the new version. Although some differences between the current and the new FESA version exist, no major problems are visible at the moment.

The next important development will be the DAQ for the beam position monitoring in the UNILAC at a cycle frequency of 50 Hz to demonstrate the real-time performance of FESA.

REFERENCES

[1] T. Hoffmann, "FESA – The Front-End Software Architecture at FAIR", PCaPAC 2008, Ljubljana, Slovenia.
[2] T. Hoffmann, M. Schwickert, G.Jansa, "FESA at FAIR", PAC 2009, Vancouver, Canada., to be published
[3] www.picmg.org xTCA for large scale physics applications, PICMG-no.: phyTCA
[4] V. Baggiolini et al, “JAPC - the Java API for Parameter Control”, ICALEPCS’2005, Geneva, Italy
[5] G. Jansa et al., "A New DAQ Installation for the SIS18 Beam Position Monitoring System at GSI", ICALEPCS2009, Kobe, Japan
[6] www.struck.de
[7] T. Hoffmann, P. Forck, D. A. Liakin, “New spill structure analysis tools for the VME based data acquisition system ABLASS at GSI”, BIW’06, Batavia (USA), p. 343-350.
[8] R. Haseitl, C. Andre, .H. Bräuning, T. Hoffmann, R. Lonsing, "Integration of Programmable Logic Controllers into the FAIR Control System using FESA", PCaPAC2010, these proceedings
[9] Frank Locci, BE/CO/FE, CERN, Switzerland
[10] http://qt.nokia.com/

FAIR TIMING MASTER

Mathias Kreider, Tibor Fleck, GSI Darmstadt, Germany

Abstract

In the scope of building the new FAIR facility, GSI will implement a new timing distribution system based on WhiteRabbit. The FAIR system will resemble a tree topology, with a single master unit on top, followed by several layers of WR switches, down to about two thousand timing receivers throughout the facility. The Timing Master will be a mixed FPGA/CPU solution, which translates physical requirements into timing events and feeds them into the WR network. Macros in the FPGA resemble a 32x multicore with a strongly reduced instruction-set, each event processor responsible for a specific part of the facility. These processors interact in real time, reacting to interlocks and conditions and ensuring determinism by parallel processing. A powerful CPU prepares the timing event sequences and provides an interface to the control system. These tables are loaded into the RAMs of each participating processor, controlling their behaviour and event output. GSI is currently working on the WR timing system in close collaboration with CERN, making this system the future of GSI/FAIR. This contribution covers technical details on the expected timing scenario, macro internals and discussion on possible future development.

INTRODUCTION

Purpose

Future GSI/FAIR facility will use timing events to control machine actions. The FAIR Timing Master will centrally generate all necessary events for the whole accelerator facility. These will be used to trigger all beam guiding components as well as all beam diagnostic measurement devices where individual event filters apply for each single front end controller. The timing receiver is integrated into the standard FAIR frontend controller used mainly for power supplies. For all other use cases, especially all beam diagnostic devices, special timing receiver interface cards will be supplied in different form factors. Typical event reaction will be direct trigger output or IRQ. Furthermore a separate high precision clock distribution system called BuTiS for RF components where highest requirements to accuracy and synchronization apply will be closely coupled to the FAIR timing system.

The WhiteRabbit Transport Layer

The future Timing System of GSI/FAIR and CERN will be based on the WhiteRabbit architecture. WR is a deterministic field bus [2], the physical system consists of a non-meshed GbE network topology, running timing services on OSI layer II. Custom switches and endpoints are used for timing measurements and the WR protocol.
WR provides phase compensation and absolute time distribution with an accuracy down to a nanosecond. Forward error correction algorithms are employed to get highest system reliability. Deterministic lag times are made possible by using Quality of Service (QoS). This makes preferring marked high priority packets possible. Since the lag time to destination is reliably known in advance, this allows machine control packets to always arrive on time.

The FAIR Timing Master

To provide an interface to the general control system of the facility, a powerful CPU handles the abstract beam production down to the creation of sequence programs for control of Event Processing Units (EPU).

Every abstract physical part of the accelerator facility like the linear accelerators, synchrotron rings and storage rings, will be represented by a dedicated timing event generator unit.

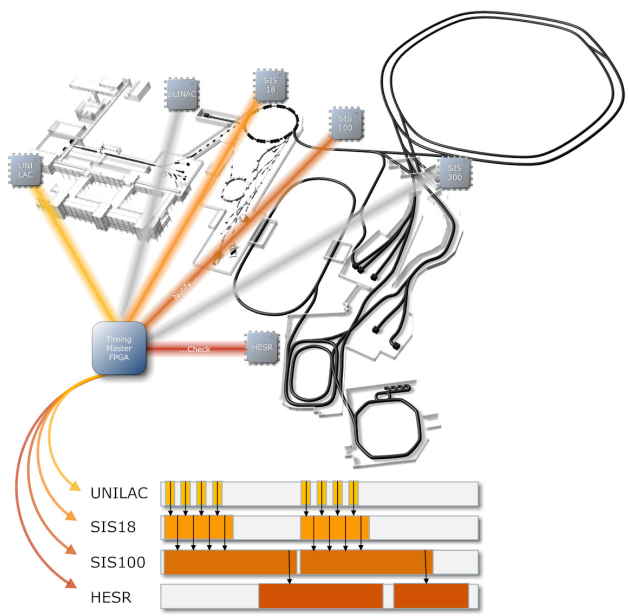


Figure 1: Mapping components to EPU programs

Interaction between these machine parts requires fast synchronisation between their timing schedules. For example, a synchrotron ring needs to time its ejections precisely with the receiving collector ring. The design therefore includes a fast mechanism for exchanges between generators.

PLANNING

The Timing Master (TM) is a mixed approach between a powerful CPU for easy integration into operating software as well as easy use of existing libraries and middleware. However, CPUs have underlying core and power management functions which make response times in the desired range unpredictable. An FPGA is used for event generation and time critical communication.

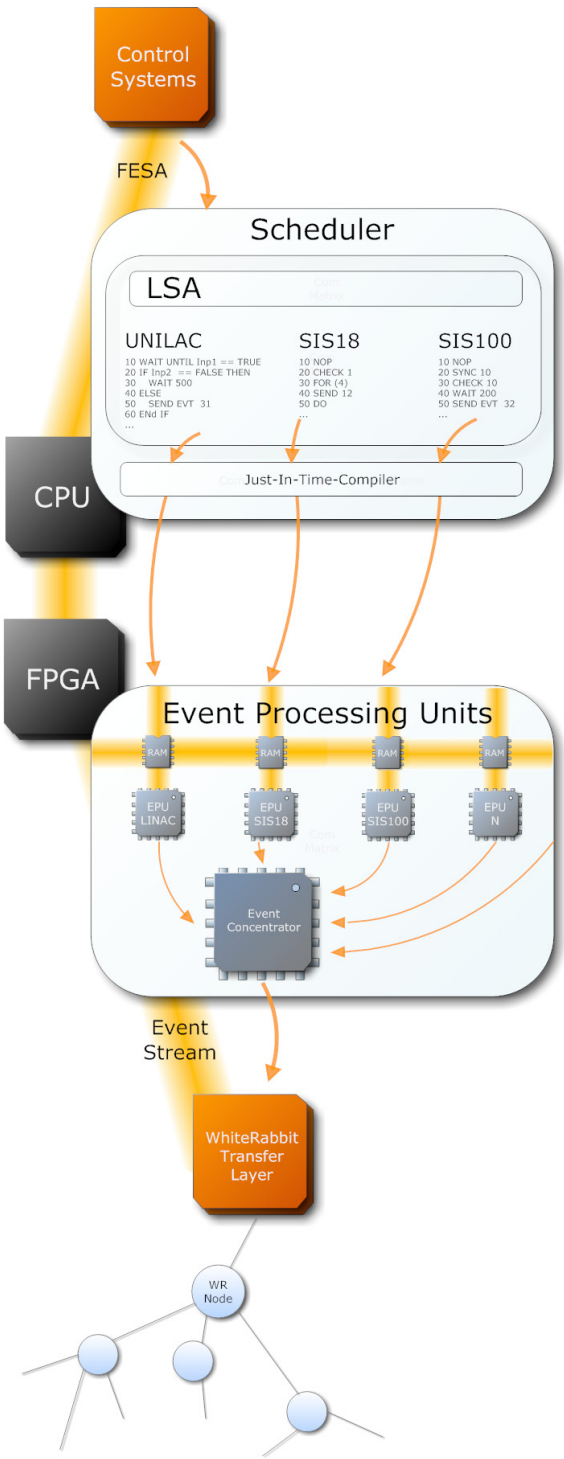


Figure 2: Timing Master Functional Blocks

ARCHITECTURE

Figure 2 shows the details of the current implementation in testing. It shows the data flow from Operating through machine translation, conversion to programmatic format and real time execution inside the FPGA submitting data to the transfer layer.

DATA FLOW

The TM's CPU gets instructions for a production line (Isotope, Amount, Energy, Source, Path, Target) from Operating. Physical requirements are translated into machine requirements by the LSA middleware. The resulting event sequences with their dependencies are transformed into event generator programs, compiled and loaded into the FPGA's memory where they will be executed.

IMPLEMENTATION

CPU

The top interface to the control system is based on FESA, a device model framework and driver package. An interface to the LSA core provides machine behavior descriptions calculated from physical parameters. Below this is the event sequencer, compiler and FPGA communications module. As soon as a production line is fully defined, relative execution time of all necessary events and dependencies between the indiv To allow reload of new programs during runtime without interruption, memory write access is managed by CPU only to write in places not currently locked for execution. Preinserted conditions in the EPUs program allow branching off to new program code on demand. Sequences for all interlock, beam abort or beam request scenarios can be predefined for all EPUs. The timing masters real-time decision logic will then always switch to safe, consistent alternative event sequences.

FPGA

The FPGA houses multiple processor macros, each with its own memory and controller. These Event Processing Units (EPUs) are all equal in implementation, their behavior is fully determined by the programs loaded into their RAM.

EPUs are made dedicated to certain parts of the facility or serve public functions, like collecting and processing beam requests from experimental stations. This has the benefit of having a human readable program for each timing generator. and interaction between involved components is easily traceable because it follows the supposed beam path. This leads to well defined modules and interfaces, which can easily be tested standalone, therefore speeding up system development.

Their programs run in parallel, are able to listen to external signals and interlocks, generate timing events and synchronise themselves with other processors by an n by n

flag matrix in a single cycle. In order to achieve fast synchronisation, the flag matrix (each EPU can signal all other EPUs) is completely realised as FlipFlops for fastest access. An EPU can set its own flag vector and read the Bit concerning itself from all other flag vectors, clearing it in the process.

EPU and Instruction Set

The EPUs opcodes define mainly programmatic courses, like jumps, branches and nested loops. They are not general purpose processors but specialised sequencers. An EPU instruction contains an Opcode, IO select instructions and dedicated data fields for event codes, time values and constants. This is not an optimal use of the FPGAs memory, but certifies execution times for each opcode and completely circumvents memory fragmentation.

WR Interface

For issues of load balancing, the Timing Master will have a 100 μs collection cycle or granularity window for outgoing events. The event concentrator macro then sends a compact stream of events to the WR module, where they are channel encoded and grouped into Ethernet packets. Packet size also has an impact on the effectiveness of the Forward Error Correction algorithm used in WR [4]. Current settings expect a packet length of at least 200 byte for the FEC to work efficiently, otherwise padding bytes must be added.

Since the Timing Master broadcasts all events facility-wide and only a few events are valid for an individual node, predefined event filters will run in each nodes FPGA. When an event is received, a node typically issues special trigger signals or interrupts.

CONCLUSION

The concept of dedicated EPUs representing accelerator components showed promise in early simulations.

A small number of EPUs were run with hand written test programs, covering scenarios with up to four cooperating EPUs. The task at hand is scaling these scenarios in simulation to copy real scenarios. As soon as the simulation is able to reproduce slowed down event sequences of the current controls system, modules will be prepared for synthesis.

OUTLOOK

A prototype system is planned to be set up in parallel to the current pulse centre in 2011. By comparing control sequences, a continuous test for aptability to the task of running the current facility can be done. First test is run with pre-written event programs, this allows testing in productive conditions without further concern about scheduling and machine calculations done above or transfer down below.

After a first design stop of the EPU macros, the next goal is an early implementation of the TMs software modules most importantly the Event Sequencer and compiler. The sequencer will be a solver tool able to synthesise the LSA output sequence by reducing it to programmatic structures and event numbers. The current compiler for the EPUs language can be converted to a JIT-Compiler module for the master.

A productive system is planned to be put into service at GSI/FAIR in 2016.

REFERENCES

[1] P. Moreira et al., “White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet”, ISPCS 2009, Brescia, Italy, Oct 2009

[2] J. Serrano et al., P. “THE WHITE RABBIT PROJECT”, TUC4 ICALEPS2009, Kobe, Japan, Oct 2009

[3] WR Switch Specifications `http://www.ohwr.org/`

[4] C. Prados Boda, T. Fleck, “ FEC in Deterministic Control Systems over Gigabit Ethernet”, THPL011 PCaPAC2010, Saskatoon, Canada, Oct 2010

FROM AN EMPTY PC TO A RUNNING CONTROL SYSTEM: A KNOPPIX LIVE-CD FOR DOOCS

G. Grygiel, DESY Hamburg, Germany

Abstract

Software deployment of operating and control systems is a hard task for beginners and can be an error prone one for experts. As an evaluation of a potential, fast deployment technique, a Linux/Knoppix Live-CD [1] for the DOOCS [2] control system software has been developed. This CD contains a DOOCS core system, some example and middle layer server programs and basic client applications. Optionally, one can install a Knoppix and DOOCS system directly from the CD. All DOOCS and operating system software are provided as Debian [3] packages. This paper will describe the Live system CD in more detail and discuss the interaction of Java Web-start based applications, other control system client applications, DOOCS name service and device servers.

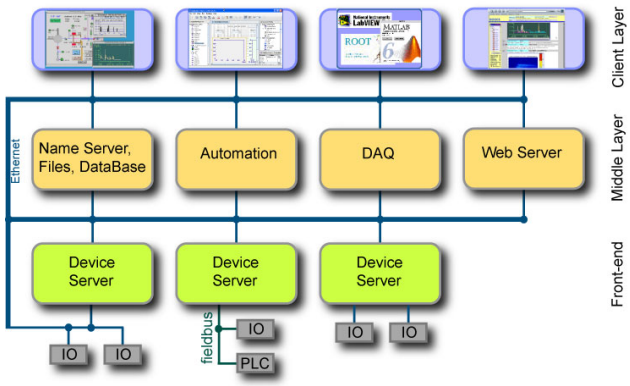


Figure 1: DOOCS Architecture

MOTIVATION

The idea is to run the DOOCS control systems with all major programs directly from a CD. The 'experts' have then an always available and workable system and this e.g. is an USB flash drive on the keychain. It's ment to provide an overview of the entire system, without complex installation and configuration. A beginner receives a fully equipped and functional system. It is possible to start immediately with the development of control system servers and having all tools at hand. The Live-CD also demonstrates the integration of the various controls system architectures, like DOOCS, EPICS [4] or TINE [5], used at modern accelerator facilities.

Features of the CD are:

- Any time, every where available.
- Quick start for beginners.
- Debug tool for experts.
- Demonstrates the whole chain, from the name service, device servers, up to the display.
- Demonstrates the interaction of the various control systems (DOOCS, TINE, EPICS).

CHOICE OF DISTRIBUTION

For almost all components, DOOCS Debian packages have been developed, therefore it should be a Debian based distribution. Currently at DESY the Ubuntu [6] distribution is used. Various tests have shown that the Live-CD made by Klaus Knopper is significantly faster than the Live-CD of Ubuntu or Debian. The KNOPPIX distribution has a very good driver support; it is fast and designed to be run directly from a CD / DVD (Live-CD) or USB stick. The first attempt to remaster a KNOPPIX live CD was immediately successful.

RECIPE TO BUILD THE LIVE-CD

Start with booting from the KNOPPIX CD. A minimum of 3 GB free disk space should be available. Then copy the complete disc to the free space. Then again one can boot the usual Linux system and start changing the content of the KNOPPIX CD. Use 'chroot' to install and configure all control system and other software. With chroot one is able to run a command or interactive shell in a special root directory. Also Internet access is possible out of the chroot environment. Change the look and feel to give the CD a personal note e.g. titles, graphics, menus. All it takes to remaster a KNOPPIX CD is described in the KNOPPIX_Remastering_Howto [7]. There are many UNIX commands to execute; therefore a good UNIX/Linux knowledge is required. It took a view interactions until everything worked and looked as expected.

To speed up the development process:

- Create the CD image.
- Start this image under qemu [8] (processor emulator) with KVM [9] support.

KVM (Kernel-based Virtual Machine) with native virtualization support helped a lot to speedup the development process. The boot up process takes less than a minute. If KVM with native virtualization support is present, it will be used by qemu automatically.

CONTENT

The CD contains a DOOCS example server (SINGENERATOR) which talks also the TINE protocol. Furthermore DOOCS, EPICS and TINE command line tools (CLI) and some graphical java programs.

In detail:

- DOOCS
  - Server programs
    - ENS (equipment name server).

        - watchdog (controls other DOOCS servers).
        - sine generator which also talks TINE
    - Client programs
        - CLI tools (doocsget, doocsput).
        - jddd [10] (Java DOOCS Data Display) talks also TINE, EPICS and Tango .
        - jDTool (Tool for displaying and changing DOOCS data.
- TINE
    - CLI tools (tget, tput, …).
    - InstantClient (Tool for displaying and changing TINE data).
- EPICS Base R3.14.11
    - CLI tools (caget, caput).
    - Server (excas).

A complete development environment for creating your DOOCS server is also available. In addition, the original content of the KNOPPIX CD is available (MPlayer, Internet access software, Mozilla Firefox and Thunderbird, GIMP, Open Office and a lot more). The latest version can be downloaded from http://doocs.desy.de/.

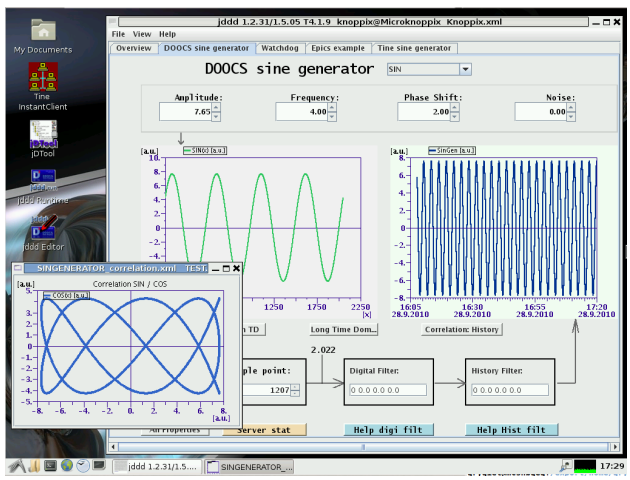


Figure 2: Knoppix with DOOCS Singenerator

BUILD A DOOCS SERVER

- Enter the following commands in a terminal:
  cd doocs/source/server/test/example
  make
- And run it:
  /home/knoppix/doocs/Linux/obj/server/test/example/example_server
- Try to change the files eq_example.h and example_rpc_server.cc, Add a further D_float property.
- Create an operator panel with jddd.

JAVA CLIENT PROGRAMS

The control system client programs are mainly java based. JavaWS (Java Web Start) is a clever mechanism to start java programs. JavaWS guarantees that you are always runs the latest version of the application and it eliminates complicated installation or upgrade procedures. The disadvantage is the dependence on a functioning Internet connection. That is why all Java programs are installed directly on the CD; it does not depend on the network to use the CD.

GOODIES

- Explore the DOOCS system and its capabilities by using the ready-to-go runtime version of DOOCS and TINE/EPICS/TANGO clients.
- Build your own DOOCS server and run it.
- Build your own DOOCS client as graphical user interface using the JDDD framework.
- Connect to the internet to browse the web, read mail, and ...
- Change your environment to access extern control systems
    - DOOCS ENS host:
      set ENSHOST
    - EPICS gateway:
      set EPICS_CA_ADDR_LIST
    - TINE:
      set TINE_HOME
      unset TINE_STANDALONE

REFERENCES

[1] KNOPPIX http://www.knopper.net/knoppix/
[2] DOOCS http://doocs.desy.de/
[3] Debian http://www.debian.org/
[4] EPICS http://www.aps.anl.gov/epics/
[5] TINE http://tine.desy.de/
[6] Ubuntu http://www.ubuntu.com/
[7] Knoppix_Remastering_Howto
http://www.knoppix.net/wiki/Knoppix_Remastering_Howto
[8] qemu http://wiki.qemu.org/Main_Page
[9] KVM http://www.linux-kvm.org/page/Main_Page
[10] jddd http://jddd.desy.de/

CONSOLIDATING THE FLASH LLRF SYSTEM USING DOOCS STANDARD SERVER AND THE FLASH DAQ

O. Hensler, W. Koprek, H. Schlarb, V. Ayvazyan, C. Schmidt, DESY, Hamburg, Germany
Q. Geng, SLAC, Menlo Park, CA, U.S.A.

Abstract

Over the last years the LLRF group developed many different flavors of hardware to control the RF systems at the Free Electron Laser in Hamburg (FLASH). This led to a variety of firmware versions as well as control system programs and display panels.
A joined attempt of the LLRF and the controls group was made over the last year to consolidate hardware, improve the firmware and develop one DOOCS front-end server for all 6 RF stations. Furthermore, DOOCS standard server are used for automation, like simple state machines, and the FLASH DAQ for bunch-to-bunch monitoring tasks, e.g. quench-detection.

An outlook of new developments for the upcoming European XFEL, using xTCA technologies, will be given.

INTRODUCTION

Over the last 15 years FLASH has evolved from a small test facility with a gun and one 8 cavity-accelerator module, running at about 100 MeV, to a photon science user facility. After the last shutdown in 2009/10 FLASH has been upgraded to 7 accelerator modules with eight 1.3 GHz cavities each, plus a 3rd harmonic module with four 3.9 GHz cavities. This set-up allows FLASH to run at a maximum beam energy of about 1.2 GeV. Presently, six RF stations are required to supply the gun, the 3rd harmonic- and the seven 1.3 GHz modules with RF.

Over this long period, the controls for the Low-Level RF (LLRF) evolved alongside the modifications of the accelerator. Many different flavours of LLRF controller hardware, starting from a pure analogue-based system for the first gun, a successfully used DSP[1] system for the modules and different versions of Simcon and SimconDSP[2] systems were developed. All these systems came with dedicated firmware, device server software and operator display panels, leading to a very inhomogeneous, global control system. Such a system was hard to maintain and applying global automation procedures was very difficult, because of the different structure and naming convention of every device server.

The effort to consolidate the LLRF system during the last shutdown will be described.

DOOCS

The Distributed Object Oriented Control System DOOCS[2] is the leading system for the FLASH accelerator. DOOCS is a standard client/server control system and based on an object-oriented approach at the front-end/server and client/display side. It is mainly implemented in C++, but there is now a Java client-side implementation called jDOOCS, on which the new display tool jDDD[3] is based. An interface for MATLAB clients is provided. The communication protocol is based on ONC Remote Procedure Calls (RPC), but a strong effort is on the way to replace them by the TINE[2] protocol.

HARDWARE

In order to achieve a homogeneous LLRF system, it is very important to start at the hardware level already. It was decided to use only two types of SimconDSP VME boards, which are equipped with ten 14 bit ADCs. One type has a Virtex V50 FPGA from Xilinx installed, which is suitable to run all control algorithms needed and is used as master card. If only additional analogue I/O is required, a SimconDSP board, equipped with a Virtex V40 is used as a slave card. The two boards are interconnected via 1 Gb fibre link to exchange the real-time data.[4]

FIRMWARE

After coming up with a common hardware platform, only a few different version of the FPGA firmware are needed, which have many parts in common, like the VME interface structure. The VME part has been optimized to allow the new 10 Hz operation of FLASH. A mapping file is provided for all VME register and tables allowing to change the firmware independent from the device server. The following firmware versions are needed :

- RF gun: This version is special, because the RF gun has no hardware probe signal. This has to be calculated from the forward and reflected power signals. In addition, the gun is a normal conducting cavity, which requires different control algorithms.
- Master board: This version includes all LLRF control and regulation algorithm as well as beam based feedbacks.
- Slave board: A simplified version to readout the ADCs and calculate the partial vector-sum is needed.

LLRF CONTROLLER SERVER

The LLRF controller server is the interface between the SimconDSP board and the control system and programmed using the DOOCS tool kit. The server runs

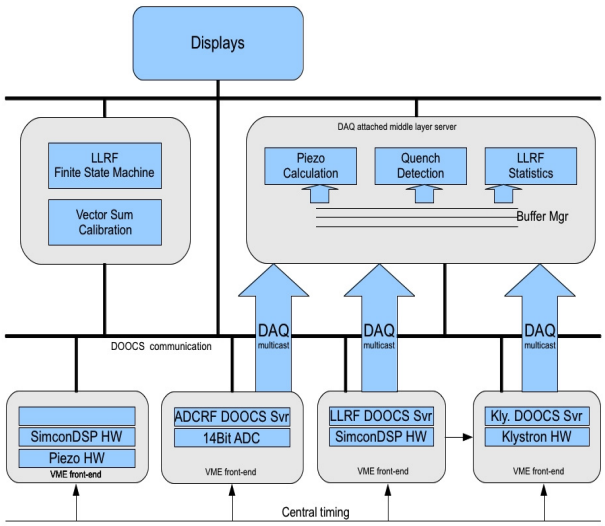


Figure 1: This picture shows the overall concept for one RF station with all required front-end computer and middle-layer server using DAQ and standard communication

on a local SPARC CPU inside a VME crate. As operating system Sun Solaris is being used. The CPU board has a hard-disk attached to store the server binary, shared libraries, configuration files, archiving and the FPGA firmware, providing complete network independent operation.

The main design goal is to have just one device server binary, which should be configurable to the needs of the individual RF stations. This was achieved by programming individual classes e.g. for cavity read-out, vector-sum, main control or board set-up. These classes are activated during start-up of the DOOCS server, while reading the server configuration file, the locations are created with its individual properties. Complex control algorithms are separated in a C library, which allows to use these algorithms in several projects and is provided by the LLRF team. All classes recover its values into the firmware after a power-up or firmware reload.

The LLRF controller server uses the FLASH nomenclature now, e.g. one location name per cavity. This eases the correlation by date with other server information and simplifies the design of operator panels. The following classes are implemented :

Board class

This class loads the FPGA firmware, in case the system is powered up or a new firmware version should be loaded. By monitoring a firmware counter, the overall operation of the firmware is monitored. In case of a failure, the RF is switched off via the FLASH timing system.

Main Class

This class is the central part of the LLRF controller server. Most of the controls, like amplitude or phase set-point, is done here. All control tables for the firmware are generated in this class and downloaded to hardware.

Due to thermal effects during startup of the accelerator, it is required to change the output rotation matrix in feed-forward mode in order to speed up. During feedback operation, these values are drifting back, though they need to be adjusted slowly. Beam based feedbacks are closely connected with the LLRF regulation and handled in this class as well.

A similar version of this class is used for the RF gun.

Cavity Class

This class reads the I and Q values of one cavity probe for one macro bunch and calculates amplitude and phase from it. A calibration parameter for each cavity is stored in this location.

Vector-Sum Class

The vector-sum class is similar to the cavity class, but is reading the partial or total vector sum of the system. The total output rotation matrix is calibrated here.

Learning Feed-Forward Class

This class monitors the error signal of the LLRF system, which is the difference between set-point and the driving output. In case this error signal gets too big, the learning feed-forward (LFF) algorithm tries to compensate by calculating new feed-forward correction tables.

Toroid Class

This class is monitoring the attached toroid signal. This channel is needed for beam-loading compensation (BLC).

Pyro Class

The Pyro signal, which allows to measure the compression in the bunch compressors, is connected to one of the ADCs. This class monitors this signal and sets a parameter needed for the pyro feedback into the firmware.

ACC1-ACC39 Class

The purpose of the 3rd harmonic module ACC39 is to linearise the 1.3 GHz RF signal. The two RF stations for ACC1 and ACC39 have to be operated in parallel. This class takes care, that amplitude or phase is set simultaneously to both stations between macro bunches.

DAQ ATTACHED SERVER

The FLASH DAQ[5] system pushes so called spectrum data (2K float array) from many front-end computer to a central shared memory with the 10 Hz repetition rate of the accelerator. This shared memory synchronizes this data on a macro-pulse basis. This allows to correlate data

from the whole machine easily. A second advantage of this scheme is, that this huge amount of data is transported only once over the network, but may be used by several DAQ attached server. A library called DOOCSddaq is provided to read spectrum or float data from the Buffer Manager directly. A trigger to the server is issued, after the data buffer are filled. This concept is used by the following LLRF server :

- Quench-detection: calculates from the I and Q values the Q loaded and detuning of each cavity. With these values a quench event can be derived. The server generates in case of a quench a flag per klystron section. This flag is used by the finite state machine to switch off the RF.
- LLRF diagnostic: this server calculates values like flattop mean, RMS, flattop slope, bunch to bunch stability and others for every cavity to generate performance statistics.
- PIEZO calculation: this server calculates the Lorenz force detuning of the individual cavities and drives the piezo front-end server accordingly.

AUTOMATION

The concept to automate the RF is based on a simple finite state machine (FSM) approach. The main purpose is to simplify the on/off-switching procedure and faster recovery from trips.

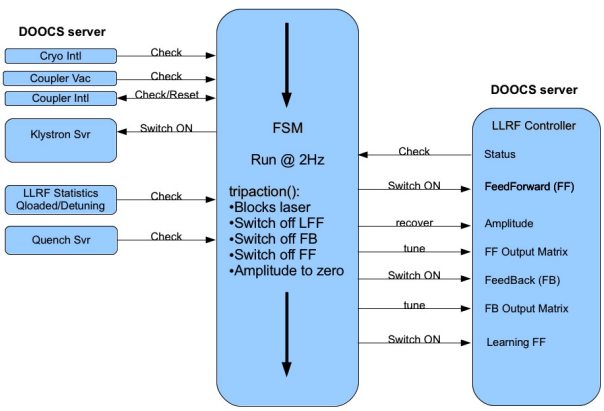


Figure 2: This picture shows the FSM concept for one RF station

This FSM is realized in the standard DOOCS server framework with the addition of the DOOCSdfsm library, which provides simple classes for monitoring float values, recover set-values or resetting interlocks. The FSM is the central server for the automation, it starts up or switches off the whole RF. All actions in other server are triggered by the FSM, giving the operator one central location to look for the status or problems of the RF system; no other software should switch the RF.

The FSM runs with a repetition rate of 2 Hz, checking several things, like interlocks, coupler vacuum, klystron status or quenches. In case of a problem in one state, the so called tripaction() function is triggered to bring the system to a save condition, then the FSM tries to recover the RF system. The same states are checked, when starting-up or recovering from trips.

OUTLOOK

For the upcoming European XFEL project, it is planned to use xTCA as hardware platform, because of the modern PCIx communication and the standardized remote monitoring capabilities. The required down-converter and fast ADCs µTCA cards are already available, the LLRF controller board is in the design phase. Porting of the LLRF controller server code from the old SPARC VME CPU to a INTEL x86 CPU is in progress. The goal is to have one source code base only by exchanging the hardware interface through compile flags. Due to the much better performance of the INTEL CPUs, it will be possible to run most of the middle layer server, like quench detection locally.

SUMMARY

The LLRF system at FLASH has been consolidated to one unified set-up for all RF stations in terms of hardware, firmware, software and naming conventions. Operator panels have been simplified and better expert panels have been designed.

The concept of a simple FSM is in standard operation, but some improvements have to be implemented to sort out conflicts between operator intervention and FSM recovery action.

The learning feed-forward algorithm has been ported from a MATLAB tool to the LLRF controller server and is in standard operation as well. Applications like vector-sum calibration or quench detection are implemented as DOOCS server already, but need more commissioning and tighter integration into the FSM framework. Further work is needed to improve the reproducibility of the RF system behaviour.

It is reasonable to say, that the first user run showed already improved performance of the LLRF controls and the new structure will be well-suited to be the base for the European XFEL.

REFERENCES

[1] DSP-Based Low LEevel RF Control as an integrated part of DOOCS Control System, V.Ayvazyan EPAC2006
[2] http://doocs.desy.de
[3] JDDD": A Java DOOCS Data Display for the XFEL E.Sombrowski, K.Rehlich, ICAEPCS 2007
[4] LLRF Control System Upgrade at FLASH, V.Ayvazyan PCaPAC2010
[5] Buffer Manager Implementation for the FLASH Data Acquisition System, V.Rybnikow, PCaPAC2008

AN ORBIT FEEDBACK FOR THE FREE ELECTRON LASER IN HAMBURG (FLASH)

R. Kammering DESY (Hamburg, Germany), John Carwardine ANL (Argonne, IL, USA)

Abstract

The lack of knowledge of the exact energy profile of the Free Electron Laser in Hamburg (FLASH) and thereby of the orbit response matrix, made the implementation of a conventional orbit feedback in the past very difficult.

The new run period started this spring after extensive modifications of the facility, showed that the responses matrixes seam now to be in good agreement with the theory, thereby allowing the application of standard orbit feedback techniques.

The physics concepts and the chosen architecture to implement such software on the middle layer and interplay with other high-level software components will be discussed. The development and implementation of this software using the DOOCS servers in combination with the dynamic components of the Java DOOCS data display (jddd) allowed a flexible and scalable implementation, which could also serve as a prototype for future implementations at e.g. the European XFEL.

MOTIVATION

The task of stabilizing beam jitter, as it is the case at most synchrotron radiation facilities, is not feasible for the FLASH linac, because the orbit can only be sampled at the maximum of pulse repetition rate of 10 Hz.

So the task of compensating fast-varying errors, for example in magnetic fields of corrector magnets or vibrations due to ground movements is here not the main focus of this orbit feedback implementation.

Instead of this the main objectives for an orbit feedback at a linear accelerator are to:

- restore saved orbits
- compensate long-term drifts
- stabilize the orbit downstream while tuning the machine further upstream
- making localized orbit changes

These are only the most important objectives an orbit feedback could attack. For FLASH it is even further envisioned to change the today practice of using individual steerers (dipole magnets) to tweak the orbit at a certain position along the machine (we will call this the *longitudinal position* in what follows), but instead of this modify beam positions using the orbit feedbacks target values at this longitudinal position.

BASIC SCHEMA OF A BEAM BASED ORBIT FEEDBACK

The basic principle of the FLASH beam based orbit feedback follows the standard techniques as e.g. described in [1]. A linear response matrix (**R**) describes the action of small changes ($\Delta\mathbf{I} = [\Delta\mathbf{h},\Delta\mathbf{v}]$) in the corrector magnet fields (dipoles) on the beam position ($\Delta\mathbf{X} = [\Delta\mathbf{x},\Delta\mathbf{y}]$) measured at the beam position monitors (BPMs).

$$\mathbf{R}\,\Delta\mathbf{I} = \Delta\mathbf{X}$$

Inverting the response matrix allows to derive the needed values to be applied to the correctors to yield a certain change in the beam position. In cases of unequal numbers of BPMs and correctors, the response matrix is non-square which can be inverted using the pseudo inverse or singular value decomposition.

$$\mathbf{I}_j = g\,\mathbf{R}^{-1}\,(\mathbf{X}_{ref} - \mathbf{X}_{meas}) + \mathbf{I}_{j-1}$$

With the gain factor g = 1 this would lead to a full correction of a given difference between the desired $\mathbf{X}_{ref}$ and actual beam position $\mathbf{X}_{meas}$, if the new current $\mathbf{I}_j$ will be written to the correctors in step j. One will usually work with a gain factor << 1 and also apply some filtering to the $\mathbf{X}_{meas}$ data to avoid ringing and overcorrection.

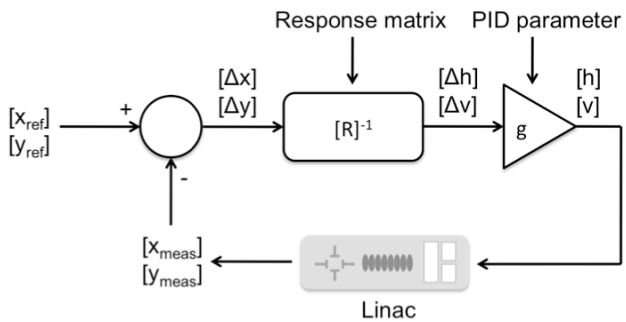


Figure 1: Basic structure of the beam based orbit feedback loop

ARCHITECTURE

The main objectives of the FLASH orbit feedback are not to damp high frequency position jitter, but more to assist operation and decouple actions within the machine.

Therefore it was planned from the beginning to implement this as a pure software feedback with moderate operation frequency (0.5-2 Hz).

The basic architecture for such a software-based feedback therefore follows the classical design of a middle layer server as described in the following section.

DOOCS as basic software infrastructure

The dominant control system at the FLASH facility is the Distributed Object Oriented Control System (DOOCS) [2]. Therefore a logical choice for the orbit feedback is to implement this software using C++ and the existing DOOCS application-programming interface (API). DOOCS offers a natural mapping of the monitors (BPMs) and correctors (steerers) to C++ objects, which significantly eases working with many devices, and thus understandablity of the code.

Optics Toolbox

In standard operation the machine optics and hence the beam transfer matrix will not change, provided the energy and quadrupole magnetic fields variations are negligibly small. Therefore we excluded the determination and manipulation of the transfer matrix from the core feedback function and instead used the well-proven optics functions from the *optics toolbox* used at FLASH [3]. This toolbox is a collection of Matlab functions providing all relevant beam optics operations needed for standard optic tasks. This toolbox is used to create inverse response matrixes and stores these in files, which are read by the orbit feedback (see Figure 2).

The orbit server

The orbit feedback server is not reading the actual beam position from the front end servers attached to the beam position monitors (BPMs), but rather it is read from a server instance (called *orbit server*) used for synchronizing and pre-processing (e.g. the intra bunch train average) as shown in Figure 2.

The orbit server itself is embedded in the FLASH data acquisition system (DAQ), from where it collects the BPM readings (for details about the DAQ system see e.g. [4]).

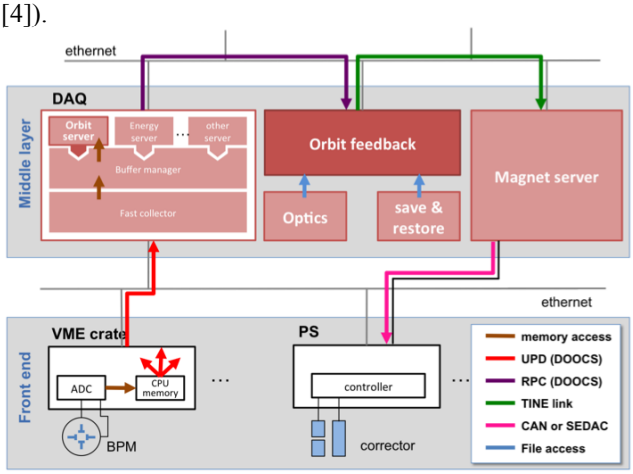


Figure 2: Architecture and data flow

The orbit feedback

The orbit feedback uses standard DOOCS RPC communication to collect BPM data and perform the calculations needed to create the corresponding vector of corrector setpoint changes. The updated corrector setpoints are written to the TINE-based magnet server, which distributes them to the relevant power supply (PS) controllers (for TINE see [5]).

The management of the reference orbits (vectors of setpoint values) will be handled through the existing FLASH Save and Restore system.

The display level

Java DOOCS data display jddd [6] is used for monitoring and control of internal states of the orbit feedback server.

Defining BPMs and steerers as DOOCS objects and subsequent mapping to jddds *dynamic lists* makes it possible to work with many devices as if they were a single instance. (FLASH, even though its length is only about 300m, contains ~ 50 BPMs and ~ 70 correctors.) This is realized by the simple technique of "draw once, use many times".

Figure 3 shows the jddd editor with the line representing a single BPM instance while in the lower right corner the same panel in run mode is showing the full list of BPMs.

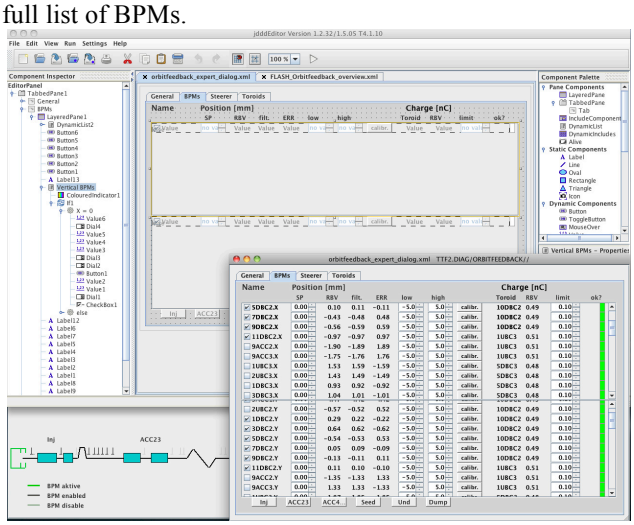


Figure 3: Java DOOCS data display, showing the BPM pane in edit and run mode

STATUS

The basic software interface for reading the beam positions and writing the corrected currents has been implemented using C++ as a standard DOOCS server. Rough estimates for total loop times (150-300 ms) have been made and show that operation with the targeted operation frequencies of 0.5–2 Hz are well suitable.

First routines for accessing response matrixes delivered by the optics toolbox have been integrated and the whole data flow chain is already operational.

All implementations have been accompanied by the continuous development of jddd panels mainly aimed for debugging, but these will also serve as a good starting point for final monitor and control panels.

First tests with the beam are planned to take place in the end of this year.

CONCLUSIONS

The implementation of a software based global orbit feedback for the FLASH facility using the existing software landscape is well on its way. The lose coupling and combination of the different control system components and protocols has allowed this new software to be developed without any need for modification of existing parts.

The combination of DOOCS object oriented approach and the dynamic generation of the displays and panels,

have proven to ease the development a lot. Such methods will be a must for working with the high device multiplicities, as one will have at e.g. the European XFEL.

REFERENCES

[1] Herman Winick, “Synchrotron Radiation Sources”, World Scientific Publishing, Singapore, 1994

[2] K. Rehlich et al, “DOOCS: an Object Oriented Control System as the integrating part for the TTF Linac”, Proceedings ICALEPCS ’97, Beijing China, 1997

[3] V. Balandin and N. Golubeva, "Current Status of the Online MatLab Toolbox for the FLASH Optics", XFEL Beam Dynamics Group Meeting, October 2007.

[4] K. Rehlich et al, “Integrating a Fast Data Acquisition System into the DOOCS Control System”, ICALEPCS’05, Geneva Switzerland, October 2005

[5] P. Bartkiewicz, P. Duval, “TINE as an accelerator control system at DESY”, Meas. Sci. Technol. 18 (2007)

[6] E. Sombrowski, A. Petrosyan, K. Rehlich, P.Tege, "jddd: A Java Doocs Data Display for the XFEL", ICALEPCS'07, Knoxville, Tennessee USA, October 2007

STATUS, APPLICABILITY AND PERSPECTIVE OF TINE-POWERED VIDEO SYSTEM, RELEASE 3

Stefan Weisse, David Melkumyan (DESY, Zeuthen)
Philip Duval (DESY, Hamburg)

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. Special effort was spent on shaping the components so that they can easily fit into small-scale but also into area-wide installations.

Here, we describe the current status of the redesigned, almost feature-complete Video System, Release 3. Individual production-level use-cases at Hasylab [1], PITZ [2] and Petra III [3] diagnostic beamline will be outlined, demonstrating the applicability at real world installations. Finally, the near and far future expectations will be presented.

Last but not least it must be mentioned that although the implementation of Release 3 is integrated into the TINE control system [4], it is modular enough so that integration into other control systems can be considered.

OVERVIEW

The origin of the featured Video System 3 (VSv3) is the Photo Injector Test Facility Zeuthen (PITZ). It is a test facility at DESY for research and development on laser driven electron sources for Free Electron Lasers (FEL) and linear colliders [5, 6].

Currently, VSv3 is almost feature-complete. Since 2008, it has emerged out of its predecessor [7], now known as Video System 2 (VSv2). The current software is a result of more than 10 years experience on video controls at particle accelerators.

As the lifetime of an accelerator facility can be a few years or decades, in contrast to the fast-pace IT world, a few design criteria should be kept in mind. Some API or operating systems can be potentially obsolete just a few years after commissioning. Both environmental considerations (radiation level) and customer demands can require frequent exchange of components and/or software evolution and upgrades. Thus there is a strong motivation to incorporate flexibility, modularity and interoperability in the design.

VSv3 was designed and implemented to meet all of these requirements, as well as those general requirements any video system must meet. These include high performance and low latency.

Selection of key characteristics/capabilities:

- raw greyscale images up to 16 bits per pixel
- raw colour images (24 bit RGB)
- integrated JPEG compression/decompression (grey and colour)
- production-level interfaces and experience in operation of: Prosilica GigE cameras, analogue cameras, JAI GigE cameras, JAI/Pulnix GigE cameras and equipment possible to attach using MS Directshow interface (Webcams etc.)
- high-bandwidth possible [8]
- low latency possible (what you steer is what you get)
- production-level 1.4 megapixel transfer, 16 bit grey, at 10 Hz update rate
- up to 30 frames per second can easily be reached
- Area of Interest (AOI)-only transfer
- shared memory interconnection of server-side components
- multicasting of video images

COMPONENTS

The video system comprises of several different components, selected ones are described in details below (see Figure 1).

The **VSv3 Transport Layer** (VSv3 TL) specifies the layout of a well-defined flexible image data type (header and bits) plus ways of transport which is integrated but not limited to TINE control system. Structure, header fields and pixel data formats are well documented.

Small Grabber Part (SGP) is the central front-end server-side component to acquire video images. To keep the C++ code simple, one SGP process will deal with only one camera at a given time. Various editions of SGP exist. Edition means it supports exactly one API to interface image sources / hardware. Most important editions at the moment are Prosilica, JAI and MS Directshow SDK, all on Windows platform. The C++ source code is kept platform independent as much as possible and references only widely available open source libraries. Thus, migration to other operating systems is expected to be on the order of hours or days. This of course depends on the availability of SDK for the chosen platform.

The connection from image source to SGP can be switched from one image source to another remotely. For example, if only two video streams are wanted in parallel, 20 cameras can be supported with just two SGP server processes. SGP provides one TINE control system output interface with VSv3 TL and one interface to shared memory (SHM).

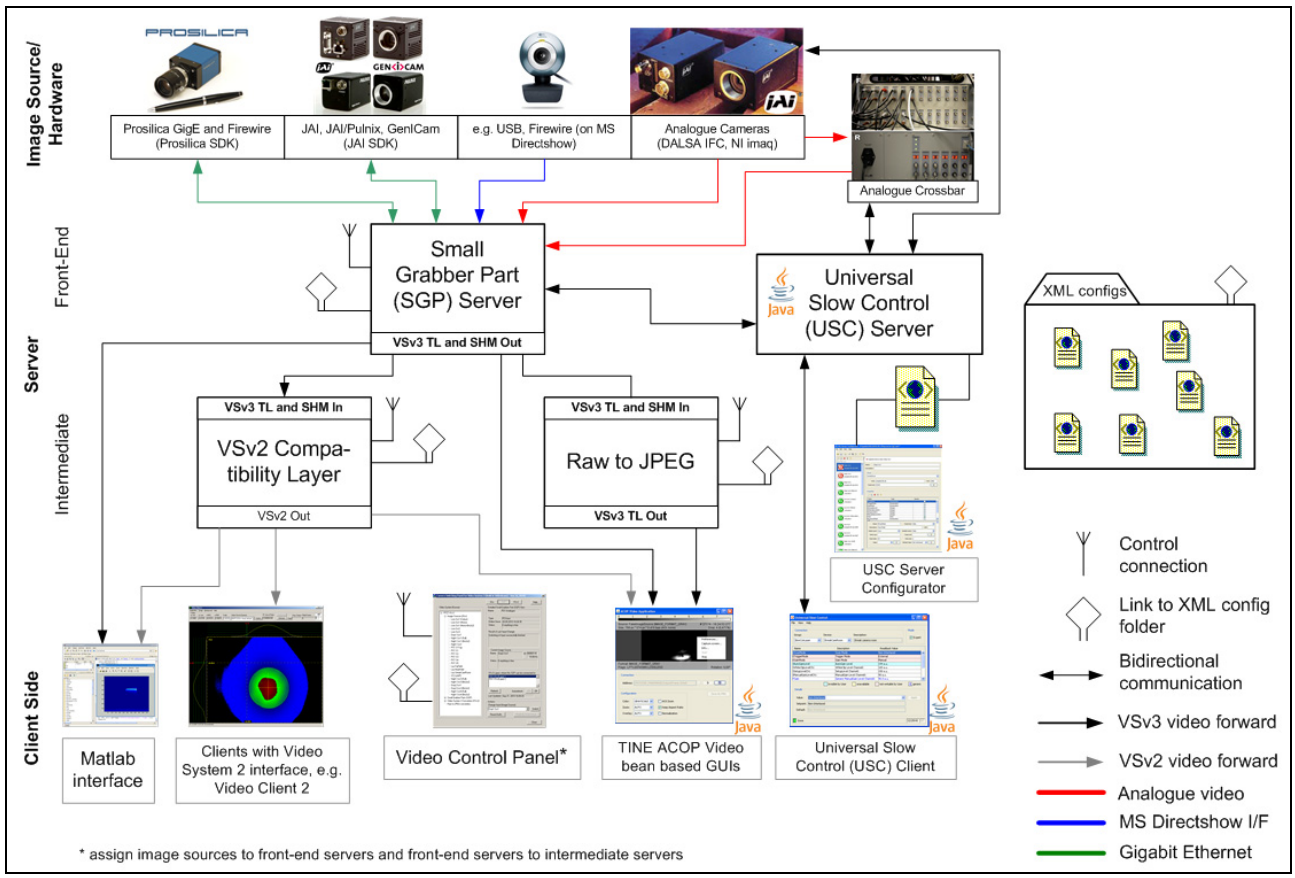


Fig. 1: Simplified layout of VSv3 components are their interaction

VSv2 Compatibility Layer is an intermediate C++ server-side component dedicated to provide backward compatibility. Its purpose is to receive image stream via VSv3 TL (using TINE or shared memory), convert the image to VSv2 format and provide VSv2 output connections (TINE and pure TCP sockets) to legacy VSv2 clients.

Raw to JPEG intermediate C++ server-side component was designed to provide easy translation of raw uncompressed images to JPEG images, with a tuneable compression factor. Input is possible via VSv3 TL (TINE or SHM), output is provided as TINE VSv3 TL. Supported are greyscale and colour images. Near-real time operation is possible. The CPU load required for this needs to be considered but resources are *easy* to provide on today's powerful commodity PC hardware.

TINE ACOP Video bean is a fundamental client-side component which displays video streams and provides basic functionality for image enhancement as well as integrated analysis made by Cosylab [9]. As Java has been selected as the target platform for future control system client-side at DESY, native Java has been used as the programming language. This gives the immediate benefit of platform independence. One might expect Java to reduce the code execution speed of the software. However, even if this does play a role (for example in low-level networking functionality), overall performance figures so far are satisfactory. With the high processing power of today's PC hardware and the periodic increases in power, Java can be considered a real alternative to native code in video system client software. As a Java bean integrated into the ACOP framework [10], it is easy to include along with other ACOP beans in Java clients (from rich clients to simple panel clients). In lieu of a dedicated client application, ACOP beans also provide a generic Video Application, which is designed to work out-of-the-box.

A well-defined **Universal Slow Control (USC) Solution** found within VSv3 provides abstraction and mechanisms to control slow parameters of hardware devices. The server part contains various connections to interface hardware, layout of parameters in hardware and well-defined TINE property interface. The USC client uses this to present hardware parameters to operators in a convenient, platform-independent Java GUI.

A **MATLAB client-side image acquisition interface** provides a simple, easy to use interface for users of Matlab. The interface supports all image features of VSv3 as well as a VSv2 input which is provided for backward compatibility. Operators are currently making good use of this interface, writing their own scripts and clients.

USE CASES

As of September 2010, most components necessary for a full-scale operation have been finished and are already installed in stable production environments at PITZ (DESY Zeuthen), Hasylab and Petra III (DESY Ham-

burg). The process of rolling out components at EMBL Hamburg has recently been started.

The Hasylab installation is focused on having many Prosilica Gigabit Ethernet (GigE) cameras all running in parallel at slow update rate (~2 Hz). Currently about 45 server processes are distributed across two machines. Images are acquired at defined positions on user beamlines in the newly built Petra III experimental hall [11]. Imaging is used for online beam centering and position monitoring. On the client side, the ACOP Video Application is used as video display. USC is used for tuning of camera's image acquisition parameters (gain, shutter speed, etc.). A special challenge has been transporting data on the 1 Gbit network interface at the server machine which is shared with the general mixed Gbit/100-Mbit controls network.

The Petra III installation consists of a VSv3 Prosilica GigE camera installation at Petra III diagnostics beamline [12] as well as an already existing VSv2 analogue camera readout which provides images of beam positions at pre-accelerators and beam distribution paths in-between. Cameras are driven with a slow update rate of about 2 Hz. On the client-side, the ACOP Video bean has been integrated into rich Java clients custom-made for Petra III control. The earlier mentioned Java-based video analysis collection of components (made by Cosylab) is a vital part of the controls setup. A special challenge at Petra III was that due to limitations in the existing control network bandwidth, certain mechanisms had to be implemented / configured in order not to exhaust limited network resources.

The PITZ setup consists of various camera types. At the moment analogue JAI cameras (M10 RS, M10 SX), Prosilica GigE (GC-1350, GC-1350C) and JAI/Pulnix GigE (RM-1405GE) are installed. Foreseen are installations of more JAI/Pulnix (RM-2030GE) and JAI GigE cameras (JAI BM141GE). In contrast to Hasylab installation, PITZ has about 25 cameras but only about 10 server processes. A camera assignment/switching panel has been provided to the operators, who use this to route video signals from source to destination. On the client side, PITZ is mainly using VSv2 software, which interfaces with VSv2 Compatibility Layer component that has been installed at server-side. VSv3 software is used directly with the VSv3-based Universal Slow Control solution for camera setup (e.g. adjusting gain and shutter in order to tune image quality at place of acquisition). Special challenges here are the demands of PITZ regarding imaging: lossless image quality, near-realtime and low latency. Furthermore constant changing of hardware and software requires a robust and flexible setup in order to avoid significant investment of time to keep it all up and running.

EMBL Hamburg has used VSv2 for sample changer monitoring and control to great satisfaction. As step by step EMBL user beamlines are commissioned at Petra III, VSv3 components are foreseen to be installed there. As a first step, an interface for Labview readout of VSv3 TL outputs has recently been provided. This is used to monitor video from Hasylab screens, which is very useful for EMBL operation.

ON THE HORIZON, PERSPECTIVE

Effort in the next months will be put on finishing intended features at the server-side. For example, applying a unique trigger event number obtained from a central source to each video frame is foreseen. Likewise, the integration of recording and playback of video sequences to Archive or DAQ installations is foreseen. At the client side, an image import/export API with stable methods to load/save the transport layer's image data type to/from a PNG file will be released, followed by an extension to sequences of images to PNG files in a ZIP container.

Over the coming years, the extension and upgrade of currently existing installations will transpire. Apart from documentation and Video System website updates, the client libraries will provide a range of APIs so that a user, no matter his software experience will be able to interface the Video System with his own tools (e.g. ROOT, MatLab, Labview, C/C++ library, Java, or .NET). VSv3 and ACOP video tools already comprise a collaboration spanning several institutes. At the same time, new collaboration partners are very welcome and are encouraged to contact us.

ACKNOWLEDGEMENTS

We would like to thank Markus Degenhardt and Gero Kube for on-site support, valuable feedback and beta-testing.

REFERENCES

[1] http://hasylab.desy.de
[2] http://pitz.desy.de
[3] http://petra3.desy.de
[4] http://tine.desy.de
[5] 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)
[6] S. Rimjaem et al., "Measurements of Transverse Projected Emittance for Different Bunch Charges at PITZ", FEL 2010, Malmö, Sweden
[7] S. Weisse et al., "Status of a versatile Video System at PITZ, DESY-2 and EMBL Hamburg", ICALEPCS 2007, Knoxville, TN, USA
[8] S. Weisse et al., "TINE Video System: Proceedings on Redesign", ICALEPCS 2009, Kobe, Japan
[9] J. Bobnar et al., "TINE/ACOP state-of-the-art video controls at Petra III", PCaPAC 2010, Saskatoon, Canada
[10] J. Bobnar et al., "The ACOP Family of Beans: A Framework Independent Approach", ICALEPCS 2007, Knoxville, TN, USA
[11] M. Degenhardt et al., "CVD Diamond Laser Alignment and X-Ray Fluorescent Screens for Petra III", SNI 2010, Berlin, Germany
[12] G. Kube et al., "Petra III Diagnostics Beamline for Emittance Measurements", IPAC 2010, Kyoto, Japan

THE FERMI@ELETTRA CCD IMAGE ACQUISITION SYSTEM

G. Gaio, F. Asnicar, L. Pivetta, G. Scalamera, Sincrotrone Trieste S.C.p.A. ELETTRA

Abstract

FERMI@Elettra is a new 4th generation light source based on a linac-driven Free Electron Laser (FEL) which is currently being built in Trieste, Italy. The CCD image acquisition system is a fundamental diagnostic tool for the commissioning of the new accelerator. It is used for the characterization and tuning of the laser, electron and photon beams. The Tango based software architecture, the soft real-time performance and the embedded image processing algorithms are described.

ACQUISITION SYSTEM

CCD

Three Basler CCD cameras (model scA780-54, scA1390-17 and scA1400-17) are currently integrated in the image acquisition system. All of them provide a Gigabit Ethernet connection and a hardware trigger input for the synchronization, and mainly differ for the number of pixels.

A total of 84 CCD cameras are installed:

- 16 are dedicated to the diagnostics of the photo-injector and seed lasers; their purpose is the measurement of the laser beam trajectory along the optical path and the characterization of the laser beam profile;
- 52 are integrated in the fluorescent screen system, which allows the analysis of the electron and photon beams along the linac and the FEL undulators;
- 16 are installed in the photon beam transport system and will be used for the measurement of the parameters of the photon beam provided to the experimental stations.

Up to 18 among the above mentioned CCD cameras have to be concurrently and continuously acquired.

Image servers

In the final configuration five server computers will take care of the acquisition of all the CCD cameras.

Each of them consists of a one-unit 19-inch rack mount server configured with two Xeon QuadCore 3.0GHz processors, 4Gb of DDR3 RAM and up to six Gigabit Ethernet links. One of them is connected to the control system network, three are dedicated to the acquisition of the CCDs and one is used for the real-time communication through the Network Reflective Memory (NRM) [1].

The servers run a GNU/Linux 2.6 kernel patched by the Xenomai real-time extension [2], which provides them with deterministic capabilities. This is used in particular by the Ethernet driver to share time-critical data among the control system computers using the NRM.

IMAGE PROCESSING

For each CCD, a Tango [3] device server is dedicated to the control of the main parameters like exposure and gain, performs the image processing and makes the results available to client applications running in the control room.

Performance and flexibility to adapt to the beam changes are the requirements that the processing software have to fulfil. The performance must guarantee to meet the deadlines because the acquisition and analysis of the image have to be done shot-by-shot. The maximum repetition rate of the linac is 50Hz, which means that a maximum of 20 ms is available to process each image. For this reason, it is convenient to analyze only the portion of image containing the beam profile, conventionally called Region Of Interest (ROI).

The image processing is divided into three steps: automatic ROI detection, calculation of the beam profile moments and data storing with a precise timestamp.

Automatic ROI Detection

Searching the beam spot inside an image could be a complicated task. Sometimes it is easier to find the parts of the image where there is no beam instead, i.e. to define the background.

In order to perform the ROI detection efficiently, the full scale image is under-sampled. The resulting samples size should be at least twice the minimum size of the beam spot in both planes in order to have at least a few points of the beam in the under-sampled image.

If necessary, the image is smoothed by a low pass filter to mitigate the presence of artifacts. A thorough design of the low pass filter parameters can dramatically enhance the magnitude of the beam profile with respect to the noise due to reflections on the vacuum pipe surface (Fig. 1).

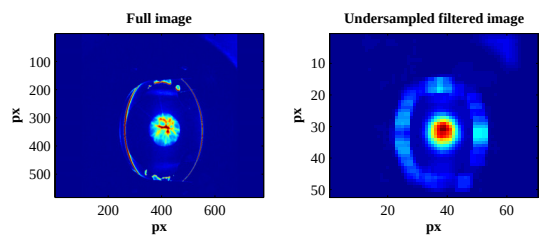


Figure 1: image of the electron beam measured at the exit of the photo-cathod gun, before (left) and after (right) the under-sampling/filtering process.

The background level is estimated through the analysis of the complementary cumulative distribution function of $P(X<=x)$, which represents the probability that a pixel value X is lower then x. This task is performed in three steps:

- subdivide the amplitude range of the image into *N* equally spaced values and store them in the array *vec_lev*[1..*N*];
- For each *vec_lev*[] element, count how many pixel values in the image are higher and store this count in the array *vec_area*[]
- Compare each element of the array *vec_area*[] with the predecessor. If the difference between *vec_area*[*n*] and *vec_area*[*n+1*] for *n*=1..*N*-1 is a above a predefined threshold *T*, then the background level is found (Fig. 2) and corresponds to *vec_lev*[*n*+1].

The experience demonstrates that for most of the beam shapes, with an 8-bit resolution image, a couple of optimal values is N=20, T=0.4.

We can assume that the pixel with maximum value in the under-sampled image corresponds to the centroid of the beam.

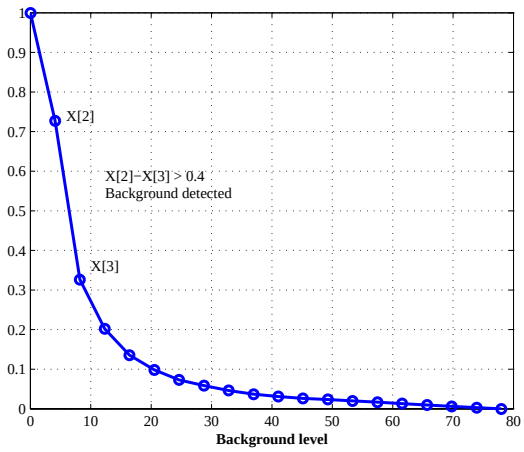


Figure 2: complementary cumulative distribution function calculated from the under-sampled image of Fig. 1

In order to find the ROI, each border of the square that initially contains the centroid is expanded until there is at least one pixel on the border which value is higher than the background level.

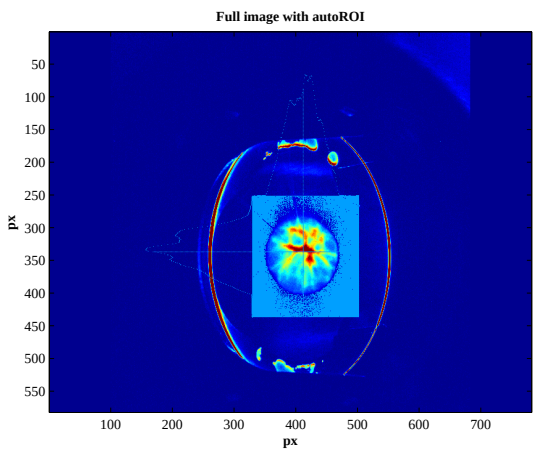


Figure 3: Final result of the image processing. ROI is highlighted directly on the image.

Once the expansion of the ROI around the beam has terminated, the coordinates of the ROI found in the under-sampled image are converted into the original image scales (Fig. 3).

Moments estimation

In order to estimate the moments of the beam profile, besides the "raw" algorithm (average and σ), three possible fitting functions can be used: gaussian, asymmetric gaussian and a seven-parameter function called "Confiteor" [4], of which the gaussian fitting function is a particular case. With the exception of the raw algorithm, the calculation of the fitting function parameters is based on the GNU Scientific Library (GSL) [5] non-linear least-squares algorithm.

A software library for the calculation of the jacobian matrix of derivatives needed in the iterative GSL algorithm has been developed. The fitting iterations stop when the predefined fitting error or a time limit is reached. The first and second moments are then analytically calculated.

The comparison of different algorithms shows that for a beam shape that is far from being gaussian, the gaussian fits could differ a lot from the correct result (Fig. 4).

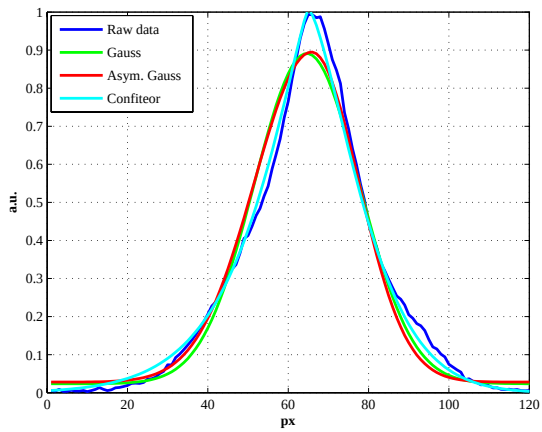


Figure 4: Comparison of the fitting functions applied to a beam which has not a gaussian distribution.

Table 1 shows the horizontal σ calculated by different algorithms and their performance. The results obtained with the raw algorithm and with "Confiteor" are in good agreement; the latter is slower but much more robust in case of "salt and pepper" noise.

Table 1: Comparison between different moment estimation methods (image with full scale size 782x582px, ROI size 120x200px)

Calculation mode	**Processing time**	**σ_x**
Raw	2.4 ms	0.309 mm
Gaussian	4.8 ms	0.255 mm
Asymmetric Gaussian	5.2 ms	0.254 mm
Confiteor	9.6 ms	0.326 mm

Data storage

The proprietary binary libraries provided by the CCD cameras vendor can only be used in the Linux user space domain , therefore it is not possible to acquire and store the CCD data in real-time. Despite this limitation, a thorough tuning of the priorities in the Tango device server and the overruling in the assignment of interrupts and processes to the eight CPU cores, allows anyway the acquisition of the beam image shot-by-shot in a reliable way.

The bunch number, a sort of time stamp which identifies each of the accelerated bunches, is distributed in real-time via the NRM along the accelerator. It is used to tag every acquired image and eventually unveil any misalignment (two images with the same bunch number) in the acquisition. The tagged images can be easily correlated with other diagnostics data (e.g. from BPMs, Charge Monitors, …) or with the strength of the accelerator magnets that can also be driven on a shot-by-shot basis.

The beam parameters calculated by the image processing are stored into circular buffers, which support both storing and retrieving operations in kernel and user space. The buffered data can be extracted specifying either the time limits or the bunch numbers. A number of filter methods (mean, median, Kalman, etc.) can be used to extract already de-noised data.

CLIENT APPLICATIONS

Graphical User Interface (GUI)

A GUI developed using Q-Tango [6] supervises the operation of the CCD cameras (Fig. 5). The graphical panel allows to deal with the CCD Tango device server API (attributes and commands) and visualizes the beam image at a selectable refresh rate and with the preferred false colour palette. It is possible to magnify the image, save a snapshot (TIFF) and store the image raw data (CSV).

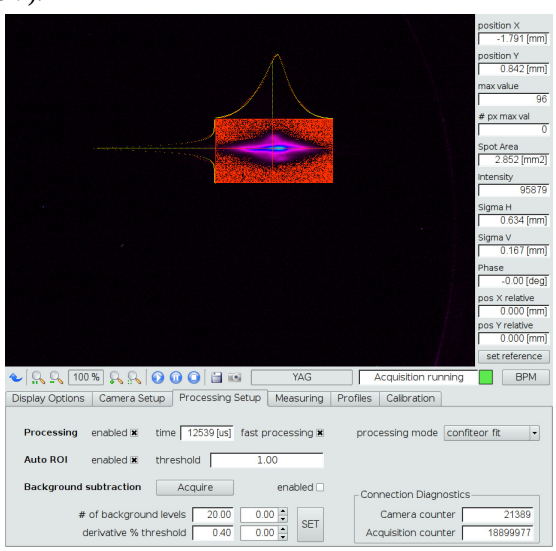


Figure 5: CCD control panel

Moreover, the panel features a smart interface for the CCD calibration process and for measuring distances in the beam image (pixel and mm).

Real-time Machine Physics Applications

The emittance is one of the most relevant parameters that must be optimized in a FEL. It could be measured by changing the focusing characteristics of a quadrupole magnet and measuring the corresponding size of the electron beam using a downstream fluorescent screen. By driving the quadrupole magnet current synchronously to the acquisition of the screen images, it is possible to obtain a good estimation of the emittance in less than 20 shots [7].

Another measurement that takes advantage of the shot-by-shot acquisition is the evaluation of the timing jitter of the electron bunches. The measurement consists in titling each electron bunch by means of a RF deflecting cavity with the proper phase, and intercepting it with a fluorescent screen. The projected image acquired by the CCD represents the longitudinal profile of the bunch and the movement of the centroid in the vertical axis corresponds to the beam timing jitter.

CONCLUSION

The CCD camera acquisition system is one of the most important diagnostics tools for the commissioning of the new accelerator. The capability to acquire beam images and correlate them with the other machine parameters on a shot-by-shot basis has contributed significantly to the success of the commissioning operations.

The auto-tuning of the image processing algorithms, the optimization of the processing code and the introduction of an abstraction layer to allow the integration of other CCD camera vendors, are some of the new developments foreseen for the future.

REFERENCES

[1] M. Lonza et al., “Beam-based feedbacks for the FERMI@Elettra Free Electron Laser”, IPAC’10, Kyoto, May 2010.
[2] http://www.xenomai.org
[3] http://www.tango-controls.org
[4] A. Lutman, Private communications, May 2010.
[5] http://www.gnu.org/software/gsl/
[6] G. Strangolino et al., “QTango: a Library for Easy Tango Based GUIs Development”, ICALEPCS’09, Kobe, October 2009.
[7] S. Di Mitri et al., “Recent Commisioning Experience on the FERMI@Elettra First Bunch Compressor Area: Investigations of Beam Dynamics, Modeling and Control Software”, FEL’10, Malmö, August 2010.

EPICS APPLICATIONS IN THE CONTROL OF SPES TARGET LABORATORY

M. Giacchini, A. Andrighetto, G. Bassato, N. Conforto, L. Giovannini,
INFN, Laboratori Nazionali di Legnaro, Legnaro (Padova), Italy

Abstract

The project of a new facility for the Selective Production of Exotic Species (SPES) has started at LNL. Radioactive ions will be produced by impinging an UCx target by a 70MeV, 200μA proton beam delivered by a commercial cyclotron. Then, the unstable ions will be accelerated by injecting them into the LNL superconducting LINAC. The construction of Target and Ion source prototype (Fig. 1) is at an advanced stage and, after more than two years spent in its construction, preliminary extraction tests were carried out with non-radioactive beams. The control of Target instrumentation is based on EPICS; we describe here the basic choices on hardware and software tools on both IOC and client side and give a brief description of last developments.

THE TARGET LABORATORY INSTRUMENTATION

The target instrumentation controls the beam extraction and transport up to a diagnostic station where the physical characteristics of the beam are measured. The beam production is obtained by heating the target to a temperature of about 2000 C, necessary for the optimal extraction of ionized fragments.

The power required for heating is delivered by an array of high current power supplies (LAMBDA GENESYS series) configured in a master-slave chain and providing a current in excess of 1300 A. Other heating methods are foreseen for the future (i.e. using a laser or a microwave source) but currently only the ohmic dissipation has been used. Once extracted, the beam is focused by an electrostatic lens of three quadrupoles f ed by a set of HV bipolar power supplies (a special assembly of rack mount units manufactured by Ultravolt).

The target and the power supplies are placed on an insulated platform that is brought at about 60KV from ground by a FUG (HCP series) power supply. The electrical power required by GENESIS modules is transferred to the platform through a 20KW insulation transformer. An Ethernet transceiver from copper (100Base-T) to fiber optic is used to link the control network to the instrumentation placed over the HV platform; a multi-port Ethernet to serial converter (Comtrol Device Master) is then used to connect the devices equipped with a serial interface.

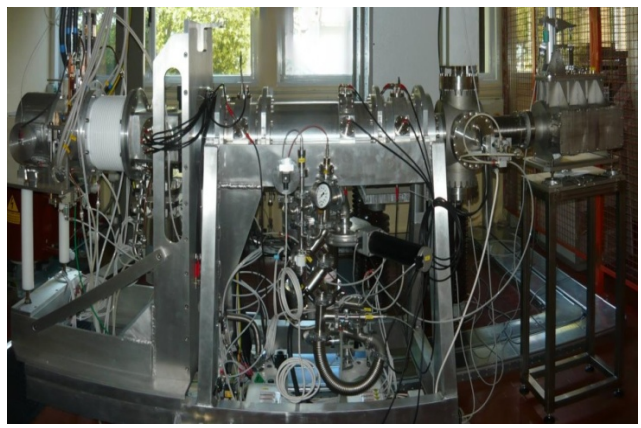


Figure 1: The target chamber and ion source

CONTROL DEVICES

The devices used to control the beam production and extraction are Linux-based IOCs. The LAMBDA-GENESYS master unit has a serial RS232 link to the host controller, which is a standard PC running on CentOS Linux. This OS distribution has been chosen because it is open-source, stable and completely compatible with RedHat. The device support is derived, with minimal modifications, from the driver developed at PSI, based on StreamDevice[1]. The HV power supplies (both Ultravolt and FUG) have an analog interface and are controlled by means of three microIOCs manufactured by Cosylab (SI).

These devices are embedded controllers based on a PC104 board and running under Debian Linux (preloaded on a flash disk). Each unit has three I/O boards, providing an adequate number of analog and digital I/O channels. EPICS drivers and debugging utilities come built-in with the controller software.

BEAM DIAGNOSTICS

A diagnostic station has been placed at the output of the electrostatic triplet to measure the beam current and profile. The beam current is measured by means of a faraday cup, while the profile is reconstructed by sampling the currents acquired by a set of horizontal and vertical grids. Stepper motors are used to insert/extract the devices along the beam line. The data acquisition system is implemented in a VME crate and runs under Vxworks.

The controller is an Emerson (formerly Motorola) MVME3100, the ADC is a XYCom XVME566 board, while the stepper motor controllers are home made devices. A modified version of the diagnostic box has been realized to measure the beam emittance: the main difference in hardware setup consists in a sliding slot moving in front of the grid array and a linear encoder to acquire the slot position. Once the raw data have been acquired, they are transferred to a host computer where the beam emittance is calculated and displayed (Fig.2).

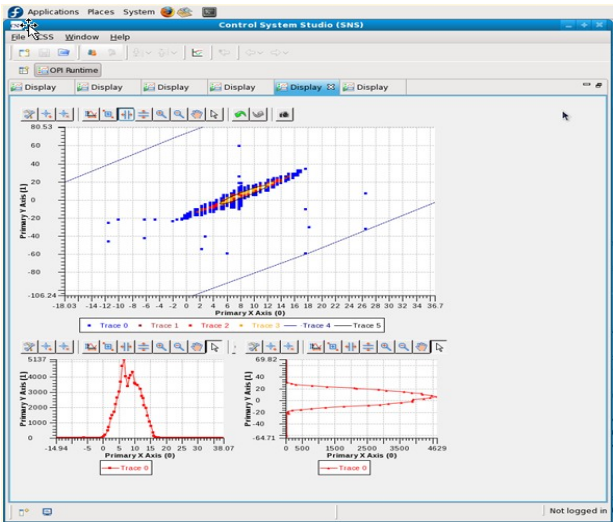


Figure 2: sample of emittance measure in a CSS screen

THE OPERATOR INTERFACE

A long term evaluation has been carried out to select the best tool to create graphic interfaces. First applications (the control of LAMBDA PS) were developed using MEDM that, despite its age, still remains an effective tool for fast prototyping and debugging. It was clear, however, it couldn't be the right choice for a project that will require maintenance over a period of twenty years at least. Then we decided to test the capabilities offered by LabView which provides a great graphic rendering and a big amount of customizable control widgets (Fig. 3).

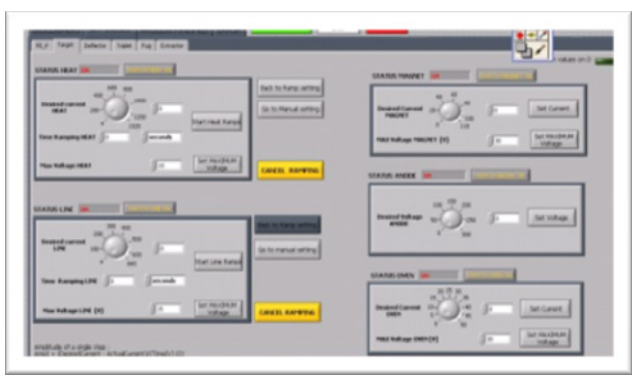


Figure 3: Control screen of HV power supplies

There are two possible approaches in using LabView as EPICS client: the most traditional is based on the "shared memory" method developed at SNS[2]; the most recent, available since the release 8.6, makes use of NI native "network shared variable" technology. We tested both, with particular attention to the SNS solution. Figure 3 shows a LabView screen of the user interface realized for HV power supplies.

At the end, however, we decided to focus on CSS (Control System Studio)[3], that resulted, by far, the most innovative and promising tool for new developments. CSS is based on Eclipse, a customizable framework, that allows the developer to extend its functionality by adding new control plug-ins. Thanks to this feature, different CSS versions are available; we decided to adopt the SNS version that includes a rich set of graphic widgets (BOY), a new implementation of Alarm Handler and a new interface to Channel Archiver.

THE ARCHIVER

The Archiver is, in an EPICS system, the basic tool for archiving and retrieving the process variables. The Archiver can work using its embedded data base or in conjunction with an external RDB. At LNL we tested both configurations, using, as external data base, the freely distributed software mySQL. Due to the limited number of PVs currently in use, the performances of mySQL are more than acceptable. However, a new project was started, in collaboration with Brookhaven National Laboratory, to study the possibility of using the Archiver in connection with the non-relational data base HyperTable [4], which is based on a novel concept of file system and exhibits very promising performances in terms of speed: this project is HyperArchiver and will be shortly presented in the next paragraph.

The HyperArchiver project

The initial idea was triggered by the observation that the most famous and fast search engine in the world (Google) makes use of a proprietary distributed data base system (BigTable) that allows managing an enormous quantity of data with a surprising efficiency and speed. Most of algorithms used for data searching are property of Google and not published but the concepts underlying the data base structure are known and can be found in other commercial DBs. One of this products, HyperTable (by Zvents), is available either in a professional version and in an open source distribution under GNU public license. We decided to test this latter version and realize a connection to the Archive engine.

We compared the store/retrieve speed with the configuration based on mySQL and it resulted that HyperTable is faster of about a factor of three in writing and more than a factor of ten in reading. A collaborative test was carried out at SNS to compare the HyperTable solution to their Archiver implementation, based on a connection with an Oracle server; also in this case it came

out that HyperTable is faster in both reading and writing, with a significant improvement in data retrieving.
We also designed the necessary plug-in to retrieve data and display them into CSS (Fig.4). The interface to CSS works but causes a significant slow-down in data reading that still has to be fixed. The collaboration on HyperTable continues and our goal is to reach a stable configuration to be used as the default Archiver installation for SPES project.

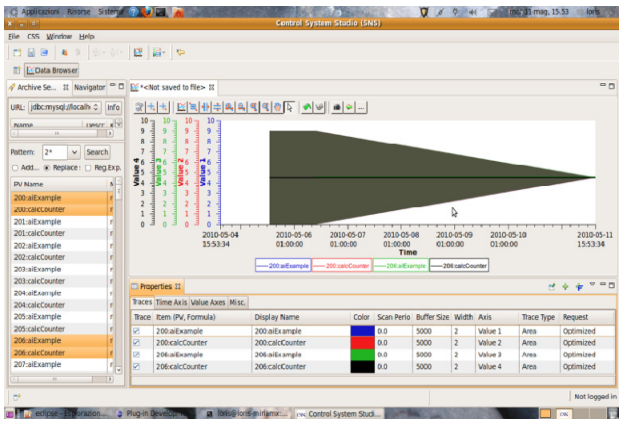


Figure 4: Snapshot of PV retrieval (simulated ramp of analog values) in CSS.

THE CONTROL NETWORK

Special care has been dedicated to the design of the control network. The following services were implemented:

- Gateway, to provide access to external services together with isolation from LNL network
- DHCP server, to manage IP addresses of control computers.
- Firewall, to protect the network from unauthorized accesses.
- Backup server, based on a Network Attached Storage (NAS) device, to allow full or incremental backup of control machines.
- Nagios [5] server, whose function is monitoring the operation of all installed IOCs and dispatching alarms in case of malfunction.
- CVS and Wiki servers: the CVS repository keeps trace of code versions and NamingConvention updating, while the Wiki server is very useful to maintain the documentation on team activity.
- PXE boot server for Vxworks and automatic reinstallation of operating system plus Epics development environment for Linux computers.

CONCLUSION

The control system of Target Laboratory has been a test bench for hardware and software technologies that will be used for SPES facility. Some technical options have been investigated enough to lead to strategic choices (i.e. using CSS for the development of user interface). Other key points must still be tested. A very important one is the integration of PLCs used for safety applications into the EPICS network. A solution based on dedicated drivers is possible for many families of PLCs, but we are strongly oriented to focus on the usage of an OPC server[6]. This approach has the considerable advantage of being independent from the PLC brand.

We also plan to continue the development of HyperArchiver, encouraged by the great interest shown for its possible application in large projects [7] where the capability of managing a huge amount of PV data in a fast way is of extreme relevance.

ACKNOWLEDGEMENTS

We sincerely thank Bob Dalesio and Robert Petkus for supporting the HyperArchiver project and for the fruitful discussions during the stage of one of authors (M. Giacchini) at Brookhaven National Laboratory. We are also grateful to Michael Davidsaver and Daron Chabot (BNL) for their valuable work in customizing the XVME566 driver [8] and writing the device support for our stepping motor controllers.

REFERENCES

[1] http://epics.web.psi.ch/software/streamdevice/
[2] http://neutrons.ornl.gov/diagnostics/documents/epics/LabVI/SNS_LabVIEWEPICS.html
[3] http://ics-web.sns.ornl.gov/css/index.html
[4] http://www.lnl.infn.it/~epics/Hypertable.pdf
[5] http://www.lnl.infn.it/~epics/NAL.html.
[6] http://www-csr.bessy.de/control/SoftDist/OPCsupport
[7] http://www-arch.iter.org/sites/epics2010/slides/
[8] http://epics.hg.sourceforge.net/hgweb/epics/xycomioc

SOFT REAL TIME CONTROL WITH CLIENT/SERVER CONTROL SYSTEM

Y. Furukawa, Spring-8/JASRI, 1-1-1 Kouto, Sayo-cho, Hyogo, JAPAN.

Abstract

Real-time properties have studied for client/server control system on single CPU system with Linux and Solaris operating system (OS) with real-time scheduler. Time jitters were within one msec for Linux OS and for Solaris OS on the MADOCA control system[1] that is the SPring-8 standard control system (CPU was 1.6GHz Intel Atom processor). These results are small enough for many synchrotron radiation experiments such as x-ray diffraction experiments with continuous scanning method. The client application can be described using scripting language, so real-time applications are developed and modified easily. The system has been used in the diffuse scattering beamline at the SPring-8.

INRODUCTION

There are many request on real time controls with msec order time resolution on synchrotron radiation experiments, such as scanning micro probe XRF, continuous scanning x-ray diffraction experiments, etc. In these applications, exact timing is not required because the counting results can be normalized by each step time or integrated intensity of incident x-ray. So the sub-msec order soft real time controls are suitable for these appllications.

To realize real-time application, real time operating system (OSs) has been used, it is, however, difficult to develop the real time applications on theses OSs because it required low-level (device driver or kernel level) software development and there are poor development support tools.

Modern OSs, like Linux or Solaris, have been improved its real time properties and became to be used for real time applications. Under these OSs, soft real time can be realized only set the framework software and these applications to use real time schedulers, such as RT-class on Solaris or FIFO and round robin scheduler on Linux.

There are many single program implementations to realize the real time properties. It requires the detailed knowledge for device control libraries and frame work, it is hard task for x-ray beamline scientist because most of them are not specialist of the control software.

If real time applications can be described using simple scripting languages, many non control specialist can develop the real time applications. It is possible if the client/server type system provides real time properties. In this paper, results of the real time property measurements in the case of the MADOCA control system on the single CPU system and it has enough for the synchrotron radiation experiments.

MEASUREMENTS OF THE REAL TIME PROPERTIES

Real time property measurements were made on Solaris 10 and Linux (vanilla kernel 2.6.34 and real time patch[2] applied kernel 2.6.33.7-rt29). In the Solaris case, parameter hires_tick=1 was set in /etc/sysconfig for 1 msec tick. For the Linux case, tickless kernel and 100Hz tick were set in kernel parameters. All the software were installed on the Atom Z530 (1.6GHz) processor based control sysmte called “Blanc-4” developed at the SPring-8[3]. The blanc-4 has 512MByte main memory and 16Gbye flash memory based storage. All the softwares were set RT-class in the Solaris case (using priocntl command) or FIFO scheduling for the both Linux case (using chrt command).

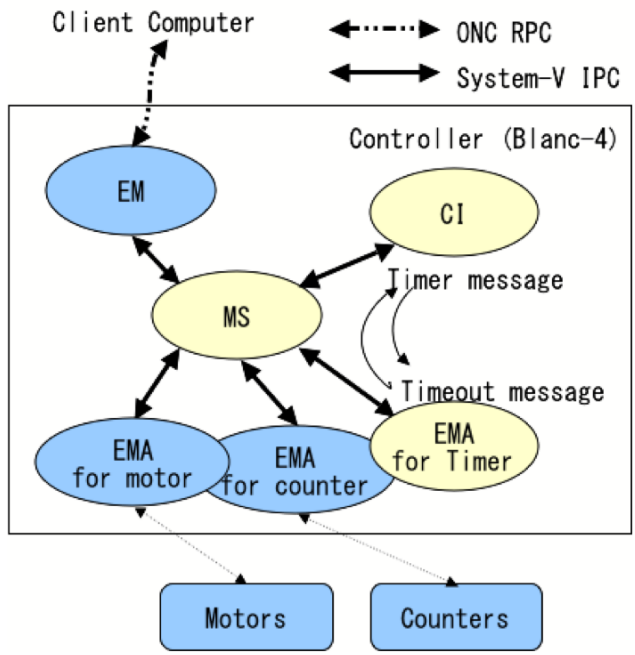


Figure 1: Software scheme of the measurements.

Software scheme based on the MADOCA control framework is shown in Fig.1. Each program communicate using system-V IPC (message queue). Command Interpreter (CI)[4], used as a client software, issued messages to the Message Server (MS). The MS transfers the control message to the Equipment Manager Agent (EMA) which controls actual devices and send back a result message to the CI via the MS. In the measurement, the EM was set as a timer, which returns a result message to the client (CI) after sleeping a given time by the message from the CI as shown in Fig. 2. The time

durations from send a message to receive the result message were measured for 1,000,000 loops.

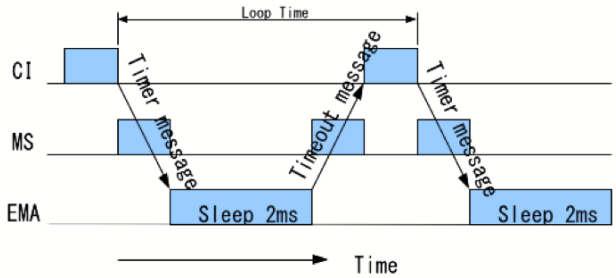


Figure 2: Time chart of measurements.

The results of time measurement are shown in Fig.3, 4 and 5 as a function of trail number for Solaris, Linux 2.6.34 and Linux 2.6.33-rt29 for first 30,000 loops. The statistics of the results were summarized in the Table 1. For the case of vanilla kernel of the Linux is not suitable for the real time applications. For the case of Solaris, time deviation is with in 0.8msec, it can be applicable some synchrotron radiation experiments. Time deviation for the Solaris 10 seems to come from SYTEM-class processes that have higher priority than RT-class processes.

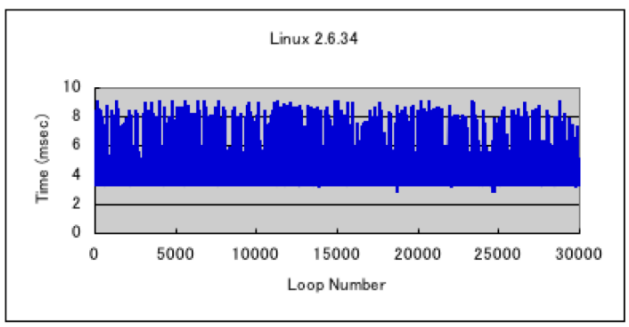


Figure 3: Result of loop time measurement for Linux-2.6.34

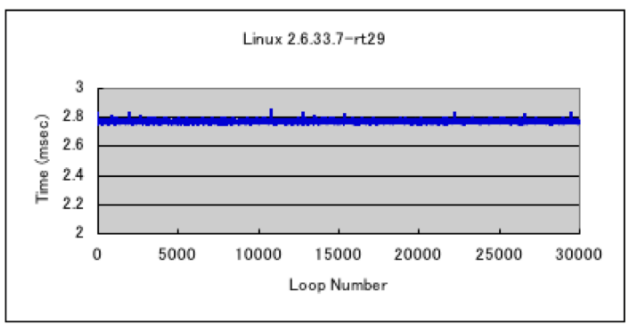


Figure 4: Result of loop time measurement for Linux-2.6.33.7-rt29

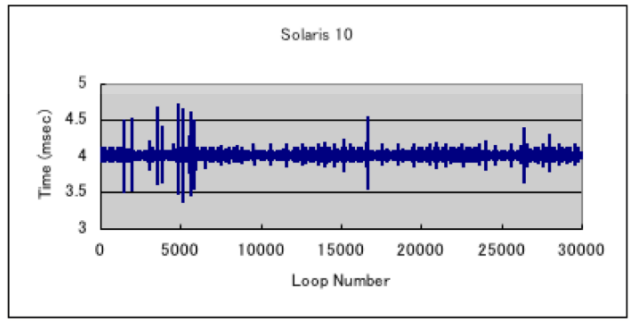


Figure 5: Result of loop time measurement for Solaris 10

Table 1: Statistics of results

OS/kernel	Meam time (msec)	Standard deviation (msec)	Min. (msec)	Max. (msec)
Linux-2.6.34	3.917	1.30	685.6	2.698
Linux-2.6.33.7-rt29	2.759	0.0091	2.731	2.969
Solaris 10	4.000	0.0175	3.349	4.417

Results for the RT-patched Linux kernel is with in 0.1msec and it is good enough for most synchrotron radiation experiments like scanning XRF, x-ray diffraction experiments. In the vanilla kernel is not pre-empt if the process is in the kernel space, while in the RT-patched kernel, the process is pre-empt in both kernel space and user space, so in the RT-patch kernel is assign the CPU time to real time process faster.

APPLICATION TO THE CONTINUOUS SCANNING X-RAY DIFFRACTION MEASUREMENT

As an application of the real-time controls, continuous scanning diffraction measurement system has been developed with Linux-2.6.33.7-rt29 system. A schematic view of the measurement system is shown in Fig.6. Diffracted X-ray by the sample is counted using x-ray detector and the detector is scanned using stepper motor. The x-ray counts are recorded as a function of the detector angle and from an analysis of the result, atomic structure is obtained.

In a conventional way, step scan was used, i.e, before counting a x-ray intensity, the detector was moved some angle. It had a dead time to waiting for end of detector motion. In continuous scanning method there is no overhead, it is, however, required msec order timing accuracy because counting duration is a few ten msec to a few seconds.

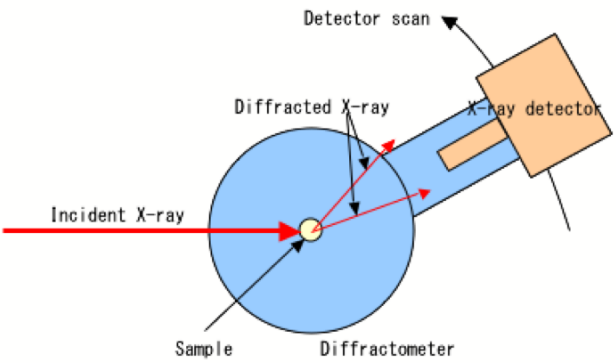


Figure 6: Schematic view of the diffration measurement.

A test of the continuous scanning was made using 1.00MHz clock instead of the x-ray counting, so the timing accuracy could be checked by uniformity of the counteing results. The result of the continuous scanning for 100msec step is shown in Fig.7. The speed of stepper motor rotation was 1000 pulse/sec. Deviation of the counting results is within 0.3%, this is good enough for most x-ray diffraction measurements. The 0.3% deviation of the counting data is corresponding to 0.3 msec timing deviation.

The counting result is not 100,000 but around 104000 counts, this measn the each loop time is 104 msec and it take 4 msec to obtaining motor position and counter data. This can be adjustable by changing the timer sleep time.

There are periodical spike on the counting data in the Fig.7. The period of the spike is about 1000 pulse, i.e. 1 sec. A motor position backing-up script was running at the same time, so the access racing to the stepper motor controller occurred. Under these racing condition to the device, timing deviation is small enough, less than required 1msec.

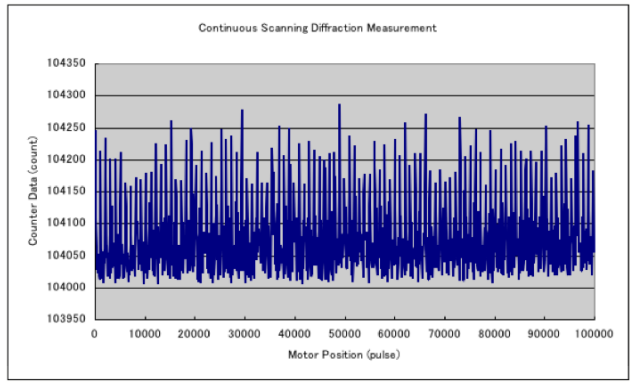


Figure 7: The result of the continuous scan for 100msec step with 1MHz input.

CONCLUSION

Real-time properties for the client/server system on Linux and Solaris OS were investigated and for Solaris 10 and RT-patched Linux case, it is shown that there are good timing accuracy. Especially for the RT-patched Linux, timing deviation is within 0.3msec.

To develop the client program, a scripting language can be used, so real-time software development becomes very easy. Note that some scripting languages invoke garbage collection and it deteriorates the real-time property. The CI is designed not to cause garbage collection.

REFERENCES

[1] R.Tanaka S. Fujiwara, T. Fukui, T. Masuda, A. Taketani, A. Yamashita, T. Wada and W. Xu, Proc of ICALEPCS'95 (1995) p.201
[2] http://www.kernel.org/pub/linux/kernel/projects/rt/
[3] M. Ishii and T. Ohata, Proc. ICALEPCS2009 (2009), p.465..
[4] Y.Furukawa, M.Ishii, T.Nakatani and T.Ohata, Proc. ICALEPCS2001 (2001), p.349

STARS ON PLC

T. Kosuge, K. Nigorikawa, KEK, Japan

Abstract

The Simple Transmission and Retrieval System (STARS) [1][2] is a message transfer software for small-scale control systems having TCP/IP sockets; STARS can work on various types of operating systems. In this study, we have successfully run the STARS server and client on the F3RP61 (Yokogawa Electric Corporation).

At present, PLCs are used for beamline interlock systems (BLISs) and PCs are used for monitoring and permission control system (CCS) of BLISs at the Photon Factory. Running STARS on a PLC makes the integration of BLIS and CCS possible. This paper provides a detailed description of the process of running STARS on a PLC.

BLIS AND CCS

Over 20 beamlines are in use at the Photon Factory and each beamline has a beamline interlock system (BLIS) for ensuring radiation safety and maintaining a vacuum environment in the beamline (Fig. 1). A PLC is used as a controller for the BLIS; it controls the beamline components (beam shutters, experimental hatches, gate valves, vacuum gases, etc.).

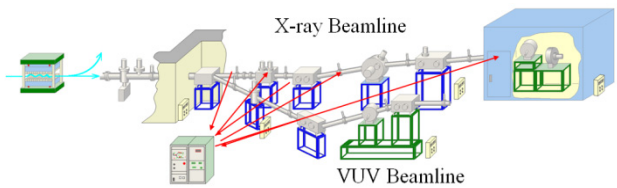


Figure 1: Beamline and BLIS.

The CCS monitors the status of BLIS and controls the permission signal, which permits beamline usage, through the PLC interface installed in each beamline (Fig. 2).

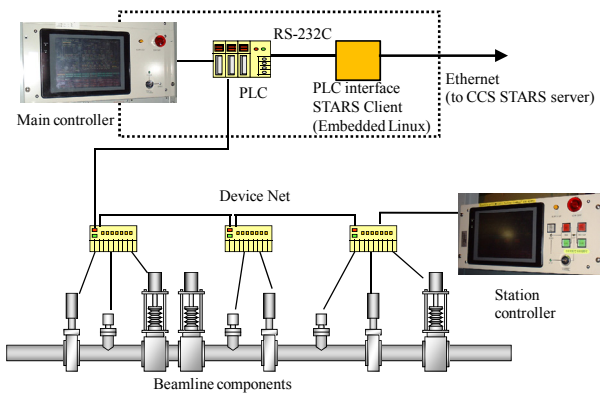


Figure 2: BLIS and PLC interface of CCS.

At present, the RS-232C is used for communication between the BLIS and PLC interfaces. The number of monitoring points that the CCS can support is limited because of the low speed of communication. Integration of the BLIS and PLC interfaces is one of the solutions to this problem.

F3RP61

F3RP61 (e-RT3 2.0/Linux) is a CPU module that can be installed on the Yokogawa FA-M3, which also has EPICS running on it [3]. In this study, we used F3RP61-2L as a test bench (Fig. 3).

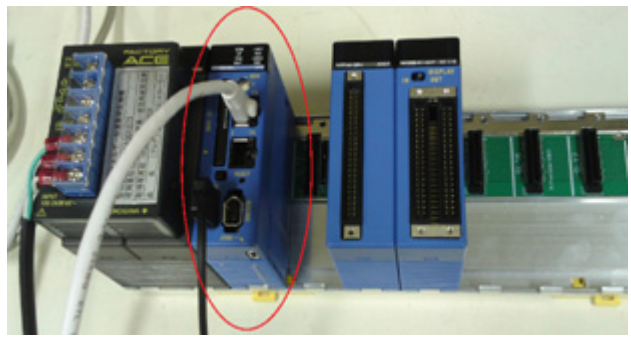


Figure 3: F3RP61 on FA-M3.

STARS

STARS is an extremely simple software for small-scale control systems having TCP/IP sockets as well as the provision for text-based message transfers (Fig. 4). A STARS server can work on various types of operating systems (the STARS server is written in Perl).

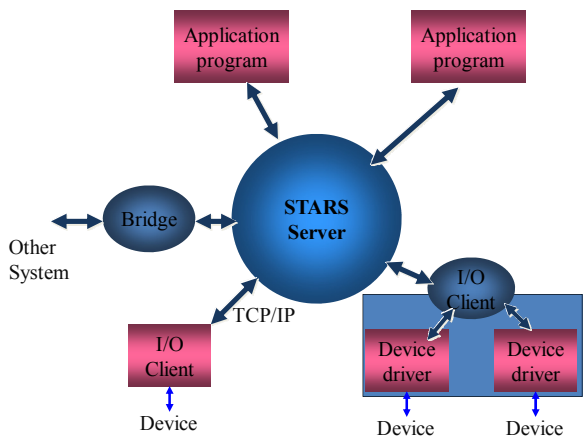


Figure 4: STARS server and clients.

STARS consists of client programs (STARS clients) and a server program (STARS server). Each client is

connected to the server via a TCP/IP socket. STARS users can upgrade the system by writing client programs, and STARS clients can participate in the system at any time without system stoppage. Recently, STARS was used for the CCS, beamline control system (see Table 1), and access control system of the experimental hall at the Photon Factory.

Table 1: Installation Status of STARS-based Beamline Control System (as on September 2010)

Category	Installed Beamline
PF-2.5GeV Ring X-ray	BL-1A, BL-3A, BL-4B, BL-5A, BL-6A, BL-6C, BL-7C, BL-8A, BL-8B, BL-9A, BL-9C, BL-12C, BL-14A, BL-17A, BL-18B
PF-2.5GeV Ring VUV and Soft X-ray	BL-2A, BL-11B, BL-13A, BL-16A, BL-19A, BL-20B
PF-AR	NE-1A, NE-3A, NW-2A, NW-10A, NW-12A, NW-14A
Other	Slow Positron Facility

STARS SERVER ON F3RP61

Various scripting languages are available for installation by means of RPM packages. In this study, we used Perl as our scripting language because the STARS server is written in Perl and therefore, it would not be necessary to modify the server program code.

STARS CLIENT ON F3RP61

STARS C Library

The Yokogawa Electric Corporation provides C libraries that enable access to IO devices or other CPUs available on the FA-M3. In addition, the C language is used for the development of a STARS client that can access IO devices available on the FA-M3.

STARS uses TCP/IP sockets and can only handle text-based messages. Skilled programmers will not find it difficult to program the STARS client. In addition, the task of programming is made easier with the availability of the STARS C library. The various functions that are part of the STARS C library are shown below.

- stars_alloc: Allocates memory for a STARS connection.
- stars_open: Opens a connection to a STARS server.
- stars_free: Releases the memory allocated for a STARS connection.
- stars_close: Closes a STARS connection.
- stars_set_timeout: Sets a time out value for the "receive" function.
- stars_get_timeout: Gets the time out value for the "receive" function.
- stars_get_handle: Gets the file handle value for a STARS connection.
- stars_send: Sends a message to a STARS server.
- stars_receive: Receives a message from the STARS server.
- stars_add_callback: Sets the function pointer for a STARS call-back function.
- stars_mainloop: Starts a call-back sequence.

STARS IO Client in C Language

A STARS client program that handles hardware is called an "IO client." The IO client waits for commands from a STARS server and executes methods on receiving such a command. Fig. 5 shows the flow chart of the IO client program.

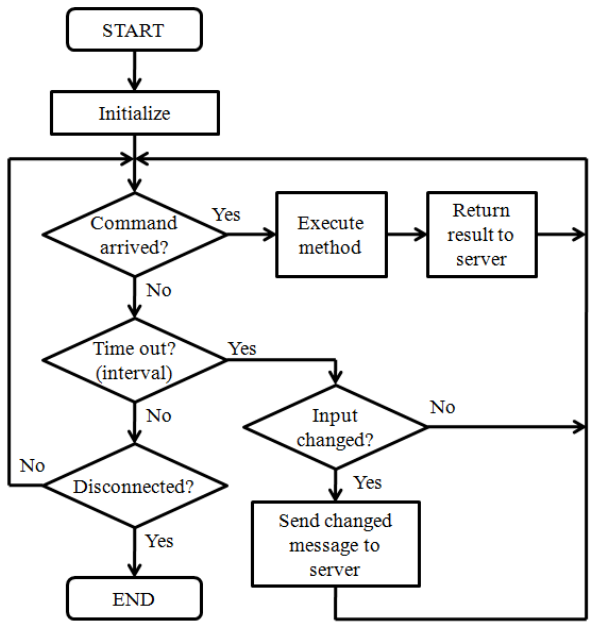


Figure 5: Flow chart of IO client.

STARS Perl Client

The STARS client program written in Perl is also available on the F3RP61. If the IO hardware of the FA-M3 cannot be accessed using the Perl program directory, then it can be accessed by the program using the STARS IO client written in C.

EXAMPLE OF APPLICATION

We have coded a simple example using the F3RP61 as a test bench (Fig. 6). A STARS server and an IO client written in C are running on the F3RP61 and a GUI is running on the PC (Windows 7 Professional).

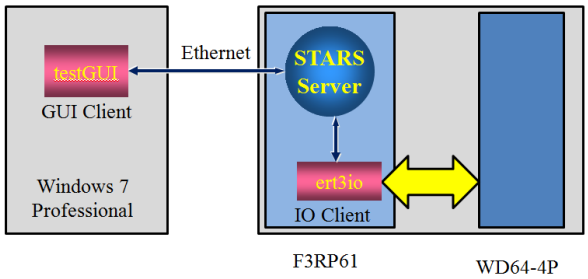


Figure 6: Overview of test bench.

When the GUI client named "testGUI" sends a command to the IO client named "ert3io" through the STARS server, the IO client executes the method that corresponds to the command and returns a result message to the GUI client through the STARS server.

The GUI is written in VB.NET and can also run on a Linux OS having MONO. Fig. 7 shows a snapshot of the GUI.

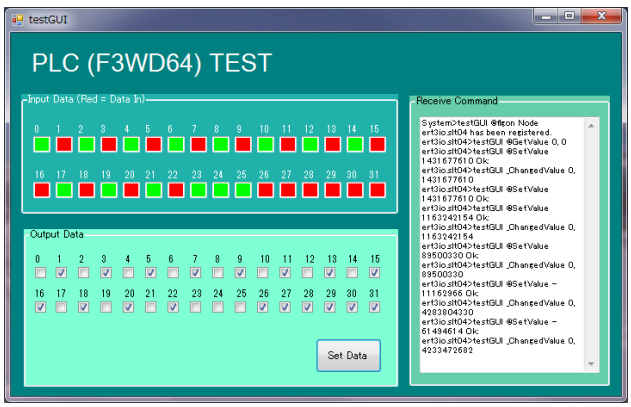


Figure 7: Snapshot of test GUI.

CONCLUSION

In this study, we have successfully run a STARS server and STARS clients on the F3RP61. In addition, we have verified that the STARS IO client written in the C language works efficiently on the F3RP61. Therefore, it can be concluded that the use of STARS on a PLC represents an effective solution for the integration of the BLIS and CCS at the Photon Factory.

REFERENCES

[1] Takashi Kosuge, Yuuki Saito, "RECENT PROGRESS OF STARS," Proceedings of PCaPAC2005, Hayama, Japan, 2005.
[2] http://strs.kek.jp
[3] J. Odagiri, et al., "APPLICATION OF EPICS ON F3RP61 TO ACCELERATOR CONTROL" Proceedings of ICALEPCS2009, Kobe, Japan.

IMPROVEMENTS FOR SIMPLE OPERATION AT SAGA-LS ACCELERATOR

Y. Iwasaki#, T. Kaneyasu, Y. Takabayashi, S. Koda, SAGA Light Source, Saga, Japan

Abstract

The SAGA Light Source is a medium-size synchrotron light research facility located at Kyusyu Island, Japan. The control system of the SAGA Light Source has been developed in the early phase of the machine commissioning. The application programs were developed using PC-LabVIEW. Commercial off-the-shelf input/output devices, such as PLC with a MS-Windows PC server, compose the input output controller with a high cost-performance ratio. ActiveX CA is used for the communication protocol between the server PCs and the client PCs. All of the components of the accelerator except the timing system are now controlled using PCs. Although the control system is stable, having many client PCs complicated the daily operation. Thus, we developed a multi-purpose client program, which is running on MS-Window 7 with a touch panel display. Furthermore, we constructed communication interface between the accelerator control system and the radiation interlock system to set the interlock mode from the accelerator control system. By using the developed multi-purpose client program and the interface to the radiation interlock system, the numbers of procedures necessary for daily accelerator operation have been significantly reduced, making the daily operation simple.

SAGA-LS CONTROL SYSTEM

The SAGA Light Source (SAGA-LS) is a medium-size synchrotron light research facility located at Kyusyu Island, Japan, and the accelerator consists of a 255 MeV injector linac and 1.4 GeV electron storage ring [1], [2]. At this time, all of the accelerator components are controlled by a digital system except for the timing system. For connectivity to the accelerator hardware, we selected commercial off-the-shelf distributed input/output (I/O) devices, such as a programmable logic controller (PLC) (Yokogawa: FA-M3) and distributed I/O controller devices (National Instruments: Fieldpoint). A difficulty at the SAGA-LS facility is its tightly restricted budget, which limits the number of staff in the facility. Thus, the control system for SAGA-LS should be simple and robust, yet inexpensive, easy to develop, and easy to maintain. One of the solutions to this problem is the use of off-the-shelf products, including PCs. The off-the-shelf I/O device and server PC works as the PC Input Output Controller (PC-IOC). Figure 1 shows a schematic view of the control layer of the SAGA-LS control system. For clarity, many of the accelerator components are omitted. We developed applications in the PC-LabVIEW environment because accelerator staffs are familiar with the PC-LabVIEW.

The PC-based control system is widely used in many facilities because of the high cost-performance ratio of using PCs. Especially recent improvements in the performance and the cost effectiveness of PCs have made them attractive for use in the accelerator control system. There are sophisticated and well-established control systems based on workstations or PC-UNIX, such as the Experimental Physics and Industrial Control System (EPICS). However, it is difficult to modify and expand the EPICS system with limited accelerator staff. Fortunately, the number of control items of the SAGA-LS is now approximately 600 and there are very few demands for real-time control. The only exception is the synchronous operation of power supplies for the four minutes of the energy ramping in the storage ring. In this case, a PLC with a preloaded ramping pattern is suitable. Hence, we designed a MS-Windows PC-based control system with off-the-shelf I/O devices [3], [4]. For the communication protocol between the server PCs and the client PCs, we used ActiveX channel access (CA) [5], which emulates the EPICS CA protocol. MySQL was adopted as the database system. Recent progress on the control system for both the linac and the insertion devices are summarized in reference [6]. The feedback control system for the magnet power supplies using external DC current transformer, feed-forward orbit, tune and coupling correction systems have been developed in past years.

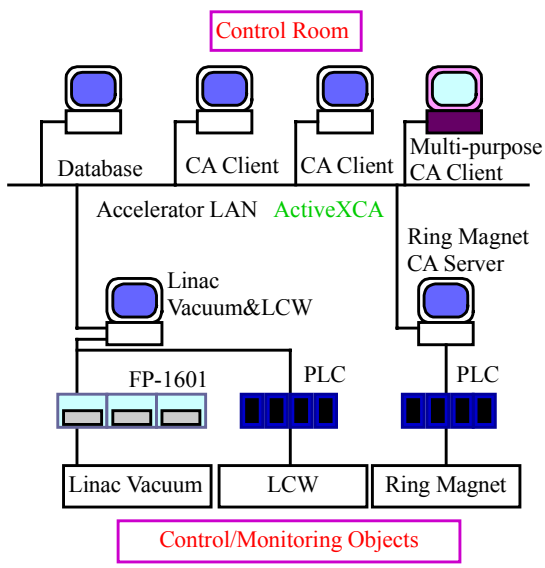


Figure 1: Schematic view of the control layer of the SAGA-LS.

#iwasaki@saga-ls.jp

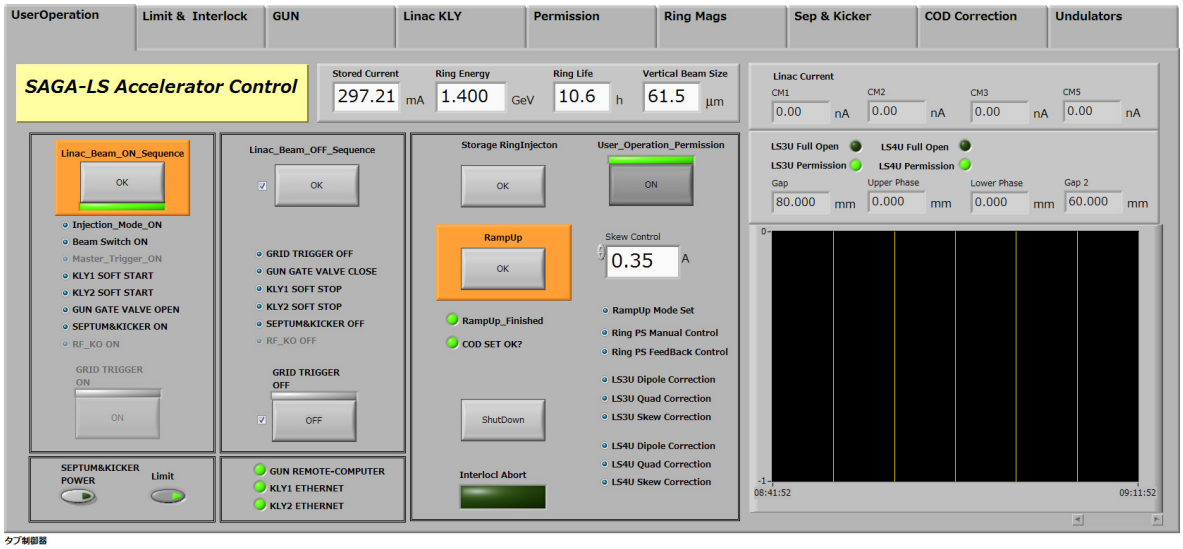


Figure 2: Front panel of the multi-purpose application program on MS-Windows 7 with touch panel. Each client function is displayed by selecting the relevant tab keys.

MULTI-PURPOSE CLIENT APPLICATION

Design Concept

As the accelerator improving, the numbers of client PCs have been growing, since each client application is created in each PC independently. Although good stability and a rigid framework of the accelerator components do not require complicated manipulation of the machine parameters in daily operation, increasing the number of client PCs increases the complexity to the operation. The necessity of setting many control knobs sometimes causes human errors. In our facility, during the injection operation, we have to carry out more than 20 processes. Although a small number of machine staff is enough for stable and safe operation in the SAGA-LS accelerator, only one accelerator staff and one assistant are actually assigned as machine operators. Thus, to avoid human errors and to simplify the machine operation, we recently constructed a multi-purpose client application program. Figure 2 shows the front panel of the application. Several operation procedures are automatically processed by the multi-purpose client application. The design concepts of the application program are as follows:

- Development in the LabVIEW environment.
- Inclusion of the major functions of each CA client program in one application program.
- Switching to these client functions by selecting the relevant tab keys.
- The application procedures are sequentially processed in the operation scheme.
- Use of MS-Windows 7 and the touch panel display.

Implementation

We have started the machine commissioning of SAGA-LS in 2004, and we used MS-Windows 2000 on the control PCs. The client programs were constructed as single-task programs for robust operation.

The construction of the multi-purpose application was made possible by recent improvements in PC CPU power and memory size. The multi-purpose client program runs on an Intel (R) Core (TM) i3 3.07 GHz CPU with 2.0 Gbyte memory and treats more than 110 EPICS CA process variables. The CPU usage is less than 10%. The application includes the electron gun, linac klystron modulator, ring power supply, global closed orbit distortion (COD) correction program, two undulators, and injection magnets (septum and kicker magnets). These client functions are switched between using tab keys, as illustrated in Figure 2. The original CA clients were constructed as “multi-stand-clients”; in other words, simultaneous and multiple runs on different PCs are possible. Such a performance is realized by using “set value” and “read back value” in ActiveX CA with LabVIEW programming [5]. Due to the “multi-clients” structure of the program, translations of the CA client programs to the multi-purpose application have become straightforward. Both the multi-purpose client program and the original client programs can be used simultaneously. In the multi-purpose application program, the “Event Structure” and “Stuck Sequence Structure” are mainly used for the automation operation processes.

The touch panel display is supported formally by Windows 7. Actually, the touch panel and the touch panel PC capability existed before Windows 7. But, with Windows 7, the high-resolution touch panel display can be used without any device driver and at low cost. We use iiyama ProLiteT2250MTS (1920x1080) for the multi-purpose client touch panel display. The resolution of the touch panel display of the system is sufficient for creating such accelerator application program. Though the touch panel display is not necessary device, it significantly increases the intuitive manipulation of the machine control.

The ring RF system, the RF-knockout (RF-KO) system for the bunch filling pattern, and the master trigger are not yet contained in this application. The master trigger system will on-line in 2011, and the RF-KO system will be added to this application in the near future. The ring RF control system will be contained with slight modification of the original ring RF CA client program.

INTERFACE TO THE RADIATION INTERLOCK SYSTEM

The radiation interlock system was originally constructed independently of the accelerator control system for rigid operation. The radiation interlock system only produces permission signals for operation of the accelerator control system at injection and ramping up. Due to the independency of these systems, the operation has become stable and the maintenance is easily performed. But, for easy accelerator operation, it is better to have an interface between the accelerator control systems and the radiation interlock system. Hence, the next five signal interface to the accelerator control system and radiation interlock system were constructed:

- Injection Mode Set/Off.
- Beam Switch On/Off.
- Acceleration of the storage ring and Accumulation Mode Set/Off.
- Acceleration Permission for the storage ring.
- Experimental Operation Permission.
- Monitoring of each status.

For constructing interface, a new PLC is installed in the accelerator LAN. By sending a signal from the multi-purpose client application to the accelerator PLC, the PLC produces the prescribed pulse to the PLC of the radiation interlock system. The status signals from the interlock system are also captured using the accelerator PLC. Figure 3 shows the communication interface between the accelerator control system and the radiation interlock system. For secure communication, the signals are hardwired and not directly connected with the Ethernet LAN.

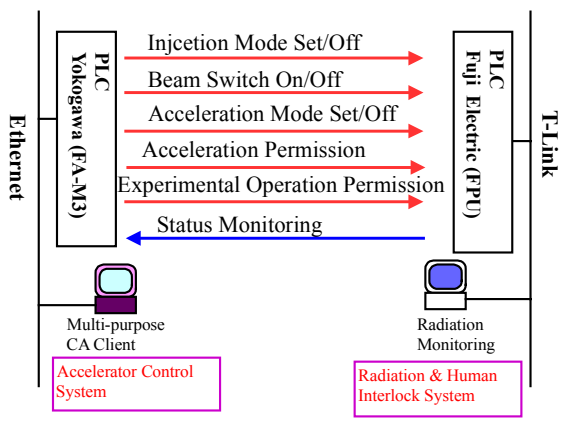


Figure 3: Interface between the accelerator control system and the radiation interlock system.

TOTAL PERFORMANCE

Before installing the multi-client application program, we had to carry out more than 20 steps from the injection to the user's experimental operation. Using the new multi-purpose application program, following eight steps for injection operation are eliminated:

I. Set Injection Mode.
II. Beam Switch ON.
III. Set Linac Klystron Shutdown.
IV. Set Ramp up Permission.
V. Set Acceleration and Accumulation Mode.
VI. Set Ring PS Tuning and Feedback ON.
VII. Set Global COD correction.
VIII. Set Insertion Devices to their home position.

In the injection processes, we only use three PCs (multi-purpose application, RF-KO, and ring RF system) and a switch (master trigger). The manipulation of the radiation interlock system is completely automated in the injection processes. Furthermore, the shutdown process was partially automated by setting the insertion device to the full open position and by setting the interlock mode of the acceleration and the accumulation to off.

By including the master trigger and the RF-KO systems on the multi-purpose application, a total of 10 steps will be reduced. We are intending to achieve "one-touch" accelerator operation by the multi-purpose client application near future.

SUMMARY

We constructed a multi-purpose client program and interfaces between the accelerator and radiation interlock system. In the multi-purpose client program, many tasks for injection are sequentially processed step by step. By developing this multi-purpose client application, the complexity of daily operation has been significantly reduced. In addition, adopting a touch panel display with MS-Windows 7 allows intuitive accelerator operation.

REFERENCES

[1] T. Tomimasu, et al., "The SAGA Synchrotron Light Source in 2003", PAC'03, Portland, May 2003, p. 902 (2003).
[2] Y. Iwasaki, et al., "Lattice Design of SAGA Synchrotron Light Source", PAC'03, Portland, May 2003, p. 3270 (2003).
[3] H. Ohgaki, et al., "Design of Control System for SAGA Synchrotron Light Source", PAC'03, Portland, May 2003, p. 2387 (2003).
[4] H. Ohgaki, et al., "PC-LabView Based Control System in SAGA-LS", PAC'05, Knoxville, May 2005, p. 3976 (2005).
[5] K.-U. Kasemir, (2003); http://icsweb.sns.ornl.gov/kasemir/axca/index.html.
[6] T. Kaneyasu, et al., "Present Status of the SAGA-LS Control System", ICALEPCS'09, Kobe, October 2009, p. 307 (2009).

CONTROL AND TIMING SYSTEM DESIGN OF CPHS *

Qiang Du#, Hui Gong, Xialing Guan, Jie Wei, Jianmin Li, Beibei Shao
Department of engineering physics, Tsinghua University, Beijing 100084, China

Abstract

The Compact Pulsed Hadron Source (CPHS) in Tsinghua University is designed as a university based comprehensive hadron research and application platform. This paper describes the control and timing system of CPHS.

INTRODUCTION

The project of CPHS in Tsinghua University consists of an accelerator front-end—a high-intensity ion source, a 3 MeV radiofrequency quadrupole linac (RFQ), and a 13 MeV drift-tube linac (DTL), a neutron target station—a beryllium target with solid methane and room-temperature water moderators/reflector, and experimental stations for neutron imaging/radiography, small-angle scattering, and proton irradiation. [1,2]

The control system of CPHS consists of an EPICS (Experimental Physics and Industrial Control System) based distributed run-time database and control system, a timing and event distribution system, and a digital low level RF control system.

The timing and event distribution system defines the global system time frame as well as specific events that trigger local devices by an event generator and receiver framework, so that the time delay of each event could be controlled in 10ns resolution, and the timing jitter of trigger signal is below 0.1ns. The hard-real-time machine protection system is also integrated in the event system so that a fault event could be responded within 50 micro-seconds. Field control signals such as water temperature, vacuum level, magnetic current, beam diagnostics, and low level RF (LLRF) phase and amplitude are monitored and controlled via the EPICS database through Ethernet.

EPICS BASED CONTROL SYSTEM

Control System General Layout

As shown in Fig 1, the EPICS control system uses several input/output controllers (IOC) to manage local process variables and establish a distributed database. The IOCs are running Linux/RTEMS kernels with device support of different local bus interfaces (serial, GPIB, stepper motor, DAQ modules, etc), communicating with local instruments monitoring and controlling water temperature, power supply, vacuum status, and LLRF status. All EPICS records are accessible from control room via Ethernet by Channel Access protocol, and are managed through Operator Interfaces (OPI) for monitoring, data logging, alarm handling, and some interlocking control. The application server and development server are responsible of providing dhcpd/bootp/nfs services for net-booting IOCs and maintaining IOC kernels, IOC applications, bootup scripts and EPICS records.

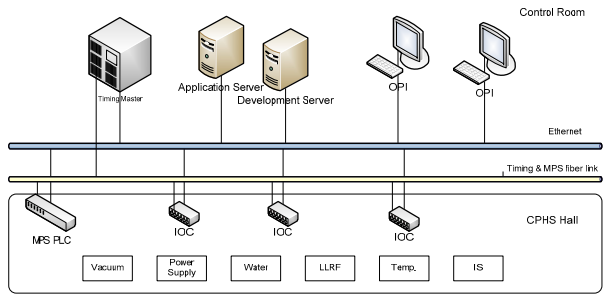


Figure 1: EPICS control system for CPHS

TIMING AND EVENT DISTRIBUTION SYSTEM

Timing System General Layout

CPHS timing events are generated, encoded and distributed through optic fiber at 108.3MHz rate (325MHz divided by 3), and then decoded by different local receivers. (Fig 3.)

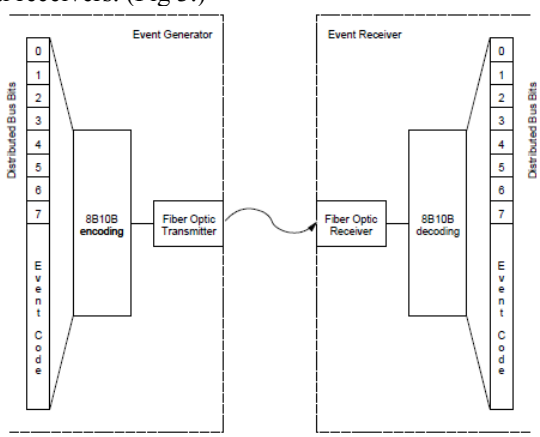


Figure 2: Event frame [3]

The event generator (EVG) is responsible of creating and sending out timing events to an array of event receivers through a fanout module. The event transfer rate is derived from the linac RF master frequency at 325MHz. The EVG is also capable of synchronizing to the AC line at 50Hz and phase delay to adjust the triggering position relative to the main voltage phase.

*Work supported by "985 Project" of the Ministry of Education of China, CAS Sciences Hundred People Initiative (KJCX2-YW-N22)
#duqiang@tsinghua.edu.cn

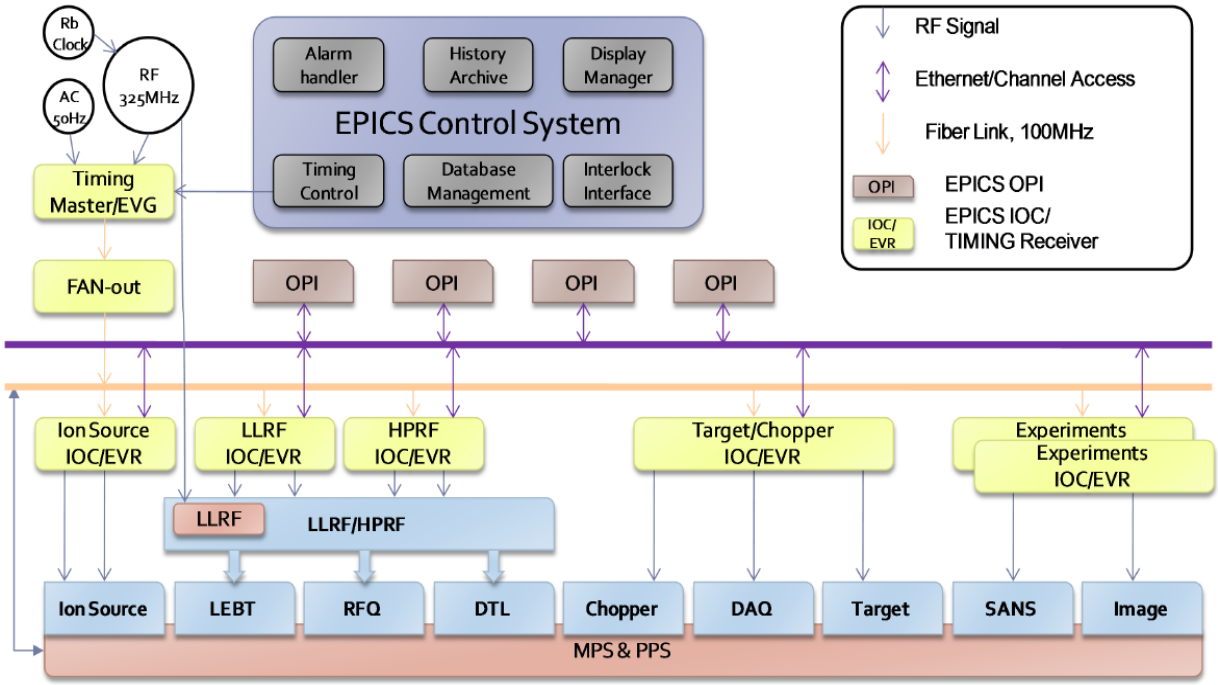


Figure 3: Framework of CPHS event distribution system

EVG accepts input from external RF clock (325MHz) with no PLL, while each EVR has a PLL tuned with ±20ppm precision to the master clock.

The event receiver (EVR) is responsible of receiving and decoding event frames. A global time stamp is also distributed as 32-bit unsigned integer to EVRs. The EVR includes a prescaler and delay counter to adjust local trigger pulse delay. The controller of EVR is integrated with an EPICS real-time IOC so that event encoder, sequence, local delay, local trigger frequency are able to be managed through any EPICS OPI.

Events are encoded and queued by EVG, and then distributed by a fanout module to local EVRs through fibre link as event frames which consists of a 16-bit frame: eight bit event code and eight bits of distributed bus bits as shown in Fig 2. The event bit rate is 20 times event code rate, which is 2.16GHz in our case.

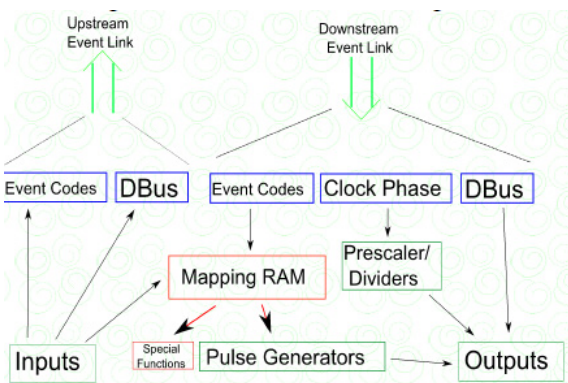


Figure 4: Downstream/Upstream event link

Besides the downstream event link, there is also an upstream from EVR to EVG with the same frame bit. This mechanism could be used as the interlock scheme for machine protection systems. (Fig 4.)

Hardware

The EVG and EVR are selected using commercial products from Microresearch Finland, which was conceptually based on event systems of ANL APS and Swiss Light Source. The hardware module is PXI compatible, and the firmware is configured with modular register mapping.

EVG and EVR are installed in separate PXI chassis with embedded controllers from National Instruments. The controllers are connected to the EPICS control network for remote access. A 12 way fanout module is used to distribute fibre signals from EVG to multiple EVRs.

The picture of EVG and EVR module is shown as Fig 5.

Software and EPICS support

There already are EPICS support for MRF hardware in use at SLS, SLAC, Diamond, etc, but the support of modular regster mapping cPCI hardware is just under development by the project mrfioc2[5,6], which also follows the regime of devLib-pci, the operating system independent device support of EPICS.[7]

Every EVG and EVR module has a separate controller running Linux or RTEMS 4.9.4 with an EPICS application. The controller is configured to be net-booted from the application server with gPXE[4] boot-loader.

The EPICS application is built on EPICS base 3.4.11 with mrfioc2 support, which containing the common devPCI driver module, MRF common PCI API, EVG/EVR device support module, and a set of EPICS records with interface of Channel Access protocol.

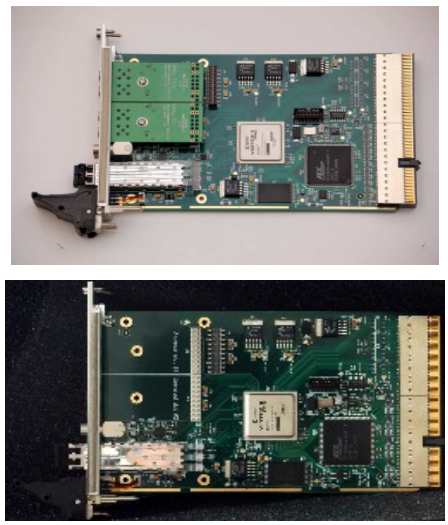


Figure 5: Picture of PXI event generator/receiver.

CONCLUSION

The prototype of CPHS control and timing system is developed with EPICS support. The timing system is built based on MRF event distribution devices, and the OS-independent EPICS device support module for EVG and EVR are tested on Linux 2.6 and RTEMS 4.9.4.

REFERENCES

[1] Jie Wei, et al, “The Compact Pulsed Hadron Source: A Design Perspective”, Journal of the Korean Physical Society, 2010, vol. 56 (1), no6, pp. 1928-1935

[2] Jie Wei, et al, “The Compact Pulsed Hadron Sourse Construction Status”, Proceedings of IPAC’10, Kyoto, Japan, 2010, 633

[3] cPCI-EVG-230 manual, Micro-Research Finland Oy, http://www.mrf.fi

[4] The Etherboot project, http://etherboot.org

[5] IOC for MRF event timing hardware, http://epics.hg.sourceforge.net/hgweb/epics/mrfioc2

[6] M Davidsaver, J Shah, E Bjorklund, “MRF Timing System IOC status”, EPICS collaboration Meeting, Aix-en-Provence, France, 2-4 June 2010

[7] MR Kraimer et al, “EPICS: Operating-System-Independent Device/Driver Support”, Proc. of ICALEPCS, 2003, 205

TINE/ACOP STATE-OF-THE-ART VIDEO CONTROLS AT PETRA III

J. Bobnar, I. Križnar, T. Kusterle, Cosylab, Ljubljana, Slovenia
D. Melkumyan, S. Weisse, DESY Zeuthen, Zeuthen, Germany
P. Duval, G. Kube, J. Wilgen, DESY, Hamburg, Germany

Abstract

The TINE/ACOP video system is a complete state-of-the-art solution for streaming beam video, featuring live analysis and live beam image display inside ACOP video component, which can be placed in any Java Swing panel. After a number of iterative improvements and embellishments, the system has matured to stable production quality in the beginning of year 2010. The system consists of the following components: a TINE device server captures a video image [1] and encodes it to the standard TINE IMAGE format. The TINE transport layer streams the IMAGE objects to clients as it would any other data chunk [2]. The Java TINE client passes the IMAGE object through the analysis Java bean, which then performs fast statistical analysis of beam position and size. The streamed image plus analysis data are displayed in the Java video component, which is part of the ACOP components. Additional capabilities are background subtraction, automatic or manual threshold subtraction, enhanced coloring and saving snapshot as PNG file. Optionally, the analysis bean can be used standalone as a common service and results are further distributed via an intermediate TINE server written in Java.

INTRODUCTION

The origin of the TINE Video System goes back to the design of the Photo Injector Test Facility Zeuthen (PITZ), which is a test facility for research and development on laser driven electron sources for Free Electron Lasers and linear colliders [1]. The optimization of an electron gun is only possible with the help of an extensive diagnostic system, including the video system.

The whole video system includes a rich set of components, covering the low level hardware integration and image grabbing, to the transport protocol and data visualization tools.

In this article we will focus on the upper level of components, which have recently been upgraded and put to use also at DESY Hamburg.

DATA ACQUISITION AND TRANSPORT

The image acquisition is implemented in a grabber server written in C++. The main purpose of this server is to acquire grayscale images from the image source and pre-process the data (e.g. compression).

The transfer of the high resolution image (up to 2 megapixels) is done using the TINE transport protocol. TINE allows various choices of data transport including multicasting, unicast UDP and TCP. Combining this with compression algorithms the TINE video system easily achieves updates at 10 frames per second.

The image transported by TINE is packed into a dedicated IMAGE data type, which is composed of an image header providing meta information about the image (frame size, bit depth etc.) and the actual image data of variable size – TINE is not limited to the transport of a fixed size image, but can be used to transfer any size one desires (within the limits of the network traffic). The IMAGE data type can also be embedded within TINE structures and is used as a standard method of exchanging image data between video system components.

IMAGE VISUALIZATION AND ANALYSIS

Java has been selected as the target platform/technology for the video system clients. The client side is responsible for visualization of the image as well as performing the data analysis and processing of the image data. In some respects we might expect Java to reduce the execution speed of the software, which would be a trade off for platform independence. This does in fact play a role regarding for instance graphics or low-level networking functionality. However, due to the high processing power of today's desktop computers, this is no longer a serious drawback and Java has proven to be very powerful and easy to use for writing the video clients.

A dedicated AcopVideo bean has been implemented, following the conventions and standards of the ACOP framework [3]. This automagically provides some common functions and tools (e.g. connection selection, drag and drop), as well as makes it easy for other developers to provide rich-clients that deal with the video.

The AcopVideo bean was implemented in pure Java, which means that it doesn't use any native resources (besides the standard ones provided by JVM) and is completely platform independent. The AcopVideo bean was designed with performance in mind, which drove the architecture and implementation of the drawing algorithm. The performance of the video bean today easily satisfies the requirements of the operation control.

In addition to high performance, the video bean provides much functionality, which is not available in the older native or commercial video clients. The AcopVideo can display any TINE video channel or still image, which can be either loaded from several standard image files (JPEG, PNG, etc.) and quality (8 to 24 bits per pixel), or provided through the TINE channel (using the event notification system in order to minimize the necessary network traffic). The AcopVideo also offers several other options for image visualization and enhancements, such as different color modes for luminosity data, histogram equalization, aspect ratio changes and zooming, display of meta information etc.

IMAGE PROCESSING AND ANALYSIS

For better understanding and easier interpretation of the image additional analysis is required. From the image of the beam one can extract the emittance as well as other properties, which the control operators are interested in. However, the extraction of such data requires a deep analysis and statistics calculation on the image data on the fly while the image is streamed from the server. Such analysis can by itself be extremely time consuming, and it can be also very difficult to perform if the beam does not have 'regular' shape (round or at least elliptical). Consequently much effort has been put into finding the most reliable and fastest analysis solution.

Statistical Analysis

The basic analysis of the image is done by calculating the statistical parameters of the beam. Using simple statistical algorithms (assuming that the beam has a non-sparse approximately elliptical shape) the mean value and standard deviation of the beam profile are calculated. The 2-dimensional analysis of the image also provides the rotational parameter of the beam ellipse.

In addition to this analysis, a side-view projection of the image is also analyzed. The pixels in a single row (and column) are summed together, what leads to two 1-dimensional profiles – one for horizontal axis and one for vertical. Similar as for the 2-dimensional analysis, the statistical parameters are also calculated for the side projections.

The calculated parameters can be used for the first approximation of the image interpretation. They provide reliable information when the image has low noise and no additional artifacts such as side light or camera pixel gain defects, split beam etc. AcopVideo bean provides functions for easy display of these parameters together with the live image; crosshair marker is used for display of the mean and standard deviation, while the side projections are plotted at the bottom and side of the image (see Figure 1). The calculated parameters can also be extracted separately and displayed for example in a dedicated table or used in further analysis.

Analysis Improvements

When the image is noisy (or generally not regular), additional algorithms have to be used to obtain better results. Thus, a Region of Interest (ROI) was introduced. When the beam is localized to a small part of the total image one can select a narrow area around the beam peak to reduce the size of the image that needs to be analyzed. Usage of the ROI significantly improves the statistical parameters, since it eliminates the contribution of the noise or other artifacts in the distant regions from the beam peak (see Figure 1).

Another improvement is the usage of a threshold value, which defines the minimum values that a pixel has to have in order to be included in the calculation. This eliminates low amplitude noise in the dark areas and puts more weight on the bright area, where the beam is located. The threshold value can be either explicitly specified by the user or calculated automatically. In the former case the threshold value is a constant in time; in the latter case the user specifies a region within the image (usually the dark region) and that region is used to calculate the mean pixel value. The mean value is then used as the threshold value during the analysis. In this case the value changes at each image update.

Next round of improvement introduced the use of the background image subtraction. A still image representing the background (the image area the beam turned off) is subtracted from each frame in order to eliminate permanent artifacts of image background. User can choose between a pre-stored image from the file system or grab a live image from the TINE channel (in the latter case the beam has to be turned off during that time in order to obtain the only background). The selected image is then subtracted from the original live image, which produces an artifact-free image used for further processing.

Further improvement of analysis was achieved by introducing a smoothing algorithm. When the image is extremely noisy, smoothing can be used to average out the noise. For each pixel the new value is calculated as the average value of a few points around the particular pixel. This can lead to more stable and reliable statistical analysis results.

Best Fit Analysis

In certain cases it turned out that even with all the aforementioned improvements, the statistical analysis still does not provide good enough results. While the mean value is approximately correct, the standard deviation might overshoot. To overcome this problem an additional algorithm has been implemented, which calculates the beam properties more precisely.

Least square curve fitting algorithm was implemented to find the best fit for the beam image side projections. We decided to use a Gauss function with linear background:

$$y = A \exp\left(\frac{(x - \bar{x})^2}{2\sigma^2}\right) + k\,x + n,$$

where

$$A, \bar{x}, \sigma, k, n$$

are the fitted parameters. To find the numerical solution to these parameters we have implemented the Levenberg-Marquardt algorithm [4]. In most cases the algorithm converges to the proper solution, but to guarantee better stability good starting values should be provided. For the first guess the statistical analysis results posed as a good guess and after the curve is fitted for the first time all consequent fits can be obtained starting with the previous results, since the beam changes are usually very slow (two consequent frames do not differ much).

This algorithm has proven to be much more reliable and trustworthy than the statistical analysis already without the use of improvements discussed in the

previous sections. If combined with the background image and threshold calculation, the algorithm produces very stable and accurate results (see Figure 1).

The drawback of the least-square fitting algorithm is that it consumes significantly more time than the statistical analysis. On a desktop PC it is still possible to observe the live image up to about 2 fps, which is in most cases enough for normal operations, but at higher rates frame drop might occur. Therefore, the user has the option to turn on or off each individual feature in order to display only the values he is interested in.

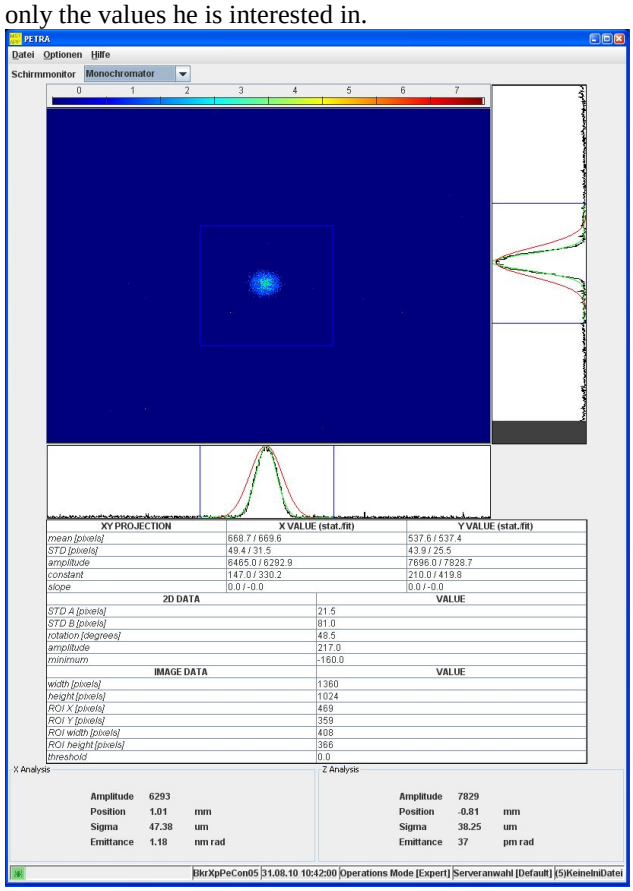


Figure 1: AcopVideo bean displaying a live beam and its profile. A region of interest is chosen around the peak of the beam (blue rectangle). Red curve is the result of statistical analysis; green curve is fitted gauss function. The table is used to show the numerical values of the analysis results.

Modularity and Analysis Server

The image analysis has been implemented independently of the video bean. This allows its usage at any level on which someone is interested into the beam analysis. The analysis bean can simply be used as an extra layer between the source of the image and the destination.

The modularity of the analysis has been used by the general analysis server, which can be used to perform the analysis instead of a desktop computer. The analysis server is written as a regular TINE server, which is registered in the TINE Equipment Naming Service and can be used as a source for the AcopVideo. All that the server requires is the TINE channel which supplies the live image and the output of the server is a dedicated data object, holding the original image and all the calculated parameters. The use of the analysis server lowers the CPU usage on the client PC, which is particularly useful when one wants to observe the analysis by several different clients (on the same or on multiple computers). However, the downside of the server is that it generates a bit more network traffic since it has to send more data (including the side view profiles etc.).

AcopVideo bean is designed in a way that it can use both the local analysis (on the client computer, where the AcopVideo is running) and the remote analysis (connected to a remote server). User is able to switch between the two options in run-time and use the one that is more appropriate at any given time.

CONCLUSION

Much progress has been done on the video system since the beginnings. A lot of effort has been put into development of high performance tools, which can be used in day-to-day operations in the control room. The recent image analysis implementation made the video applications much more than just a simple visualization tools – it became a powerful diagnostic tool for online emittance diagnostics at PETRA which tremendously helps the operators in the control room to achieve full accelerator performance.

Nevertheless, there is still a lot of room for improvements. The next step is the optimization of the transport and compression of the image, which might consequently require the optimization of the analysis algorithms. The analysis itself also leaves options for further development, such as for example 2-dimensional Gaussian fit.

REFERENCES

[1] S. Weisse et al., "Status of a versatile Video System at PITZ, DESY-2 and EMBL Hamburg", ICALEPCS 2007 Proceedings, Knoxville, TN, USA

[2] S. Weisse et al., “TINE Video System: proceedings on redesign”, ICALEPCS 2009 Proceedings, Kobe, Japan

[3] J. Bobnar et al., “The ACOP Family of Beans: A Framework Independent Approach”, ICALEPCS 2007 Proceedings, Knoxville, TN, USA

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, “Numerical Recipes in C, The art of Scientific Computing, 2nd Edition”, Cambridge University 1992

APPLICABILITY OF XAL FOR ESS

Jaka Bobnar, Cosylab, Ljubljana, Slovenia
Steve Peggs and Charles Garrett Trahern, ESS, Lund, Sweden
Todd Satogata, Jefferson Lab, Newport News, Virginia, U.S.A.
Thomas Pelaia II and Christopher K. Allen, ORNL, Oak Ridge, Tennessee, U.S.A.

Abstract

XAL is a Java-based application framework, developed at the Spallation Neutron Source (SNS). The framework is designed to provide an accelerator physics programming interface to the accelerator, and it allows creation of general-purpose applications dedicated to various parts of the accelerator.

The backbone of the XAL framework is an XML-based description of the accelerator. The XML file provides the list of all devices, their properties, and relationships between devices within the system. Since the accelerator structure is defined in the relational database, XML can be generated directly from the database using appropriate adapters. This allows the framework to be more generic and enables it to run on different sites using various configurations.

The generality of XAL and the rich set of applications and tools provided by SNS make the framework very appealing for use at other accelerator sites. The European Spallation Source (ESS) is being built in Sweden, and is similar in complexity to the SNS. XAL has therefore been considered for use at ESS for high-level applications. The applicability of XAL and prototyping for ESS are discussed in this article.

INTRODUCTION

The XAL framework was developed by SNS as a part of their accelerator physics activities. It was designed to provide a common set of tools and applications used in machine physics and accelerator control. Today the framework includes a vast set of applications such as Orbit Correction, Wire Scanner Analysis, Scanning Application etc. These applications are all used in day-to-day activities in the SNS control room.

XAL was designed from the start to be as independent from machine details as possible. Therefore a specific model was defined which provides a detailed description of the accelerator. At start-up the model is parsed and used by the framework to gain access to various parts of the accelerator. The model allows XAL use at different accelerator sites without changing the code, since the model is supplied as a set of configuration files and is the only part of the framework that needs to be adapted.

Recently XAL went under major restructuring in order to make the code even more transparent and to allow easier development of site specific applications and components. ESS, being a similar machine to SNS, appeared as a potential heavy user of this framework (now named Open XAL).

ACCELERATOR MODEL

The backbone of the XAL framework is the accelerator model. The model describes the layout of the accelerator and its parameters as they are used by the applications.

The XAL model is defined in a hierarchical structure within an XML file. This XML file is composed of several different accelerator sequences, which consist of other sequences or components each describing a particular segment of the accelerator. Combined together, they form a hierarchy of the complete accelerator down to every particular device that can affect the beam path. In addition, the XML file also provides all the necessary pieces of information required for the control of a particular physical device. For example, the magnet description includes the strength of the magnetic field, its position within the accelerator, the power supply associated with it etc. [1].

XAL uses EPICS as the underlying control system to communicate with the accelerator hardware. EPICS communication uses a single “Process Variable” (PV) as the fundamental unit for communication with high-level software via a protocol called Channel Access. Therefore, in addition to the physical description of the devices, the XML model also carries information about associations between EPICS PVs and accelerator devices. A single device can have several different PVs, each assigned to one particular device attribute.

Based on the information in the XML file, a Java model is constructed by the XAL upon start-up of an application. Each component within the XML structure is mapped to a Java device object and can be treated as such in XAL applications. Users can set or read the attributes associated with any of those devices simply by changing the value of a particular field in that object, and changes are immediately reflected in the real system through the PV registered for that particular attribute.

Though XAL currently supports only EPICS control system, the underlying mechanism is abstracted so Channel Access can be replaced by other communication protocols. This permits some aspects of XAL to be truly portable between accelerator sites.

Taking into account all the aforementioned pieces of information, one can end up with an enormous XML model, which might be very difficult to maintain. Therefore, XAL works together with the central database, which stores all the required information. A dedicated XAL application gathers the information from the database and generates the XML file. This ensures that the model is always consistent with the accelerator and

makes it very easy to implement the changes that were introduced to the physical machine, if those are registered in the database. The structure and type of the database are not prescribed, since an XML generator can be written specifically for each system. This increases portability by only requiring site-specific adaptors to extract database contents to XML. In addition, the same database can also be configured to be used as a save/restore point for control system variables allowing users to save current state of the machine and restore it at any time later or any other feature, which might need to store or load information.

ONLINE MODEL

A critical component of many beam commissioning activities involves comparison of measured quantities with model predicted values. To facilitate this, XAL includes an "online" model, which is a simple envelope tracker designed for use in applications. This model implements on-the-fly calculation of beam parameters based on the machine settings [2-4].

The online model is loosely coupled with other parts of the XAL framework. The main components are the lattice (constructed from the aforementioned accelerator model) and a probe (describing the beam and how it is to be modeled). The lattice is generated via a set of rules from the accelerator node device information (generated from the XML model) and probe information is supplied via another configuration file.

Based on this information one can use XAL to perform simple simulations of the beam behavior inside the accelerator and tweak simulated machine parameters to achieve a desired response, then use the same framework to send the desired settings to the accelerator. The online model can also be used outside of XAL, as long as the lattice and probe information are provided.

APPLICATION FRAMEWORK

The XAL Application framework is a framework for rapid development of applications with a common look and feel, which provides many features that users expect from modern applications [5].

The application framework provides a set of classes that the applications extend to use common XAL features. The framework is based on Java Swing GUI components, and provides a simple GUI builder called *Bricks*, which can be used to build XAL applications even by developers who are not experienced Java programmers.

Through the framework, developers can access various common parts such as the accelerator model, the online model etc. Putting it simply, the framework provides a complete user interface to the accelerator.

Using the XAL framework has several advantages. The most important is a consistent look and feel of all applications used by the operators and therefore, minimization of the troubles that could appear if each application had slightly different layout, menu orders, toolbars etc. Furthermore, many features (such as copy, cut & paste, printing etc.) are automatically provided, easing the load on the developer. Those features can simply be turned on or off and the developer can focus more on the application content and less on implementation details.

In addition, the Open XAL project will support localization. Each application will provide an externalized text file where all the text (menus, buttons, labels etc.) will be located. By replacing the file with a translated one, users will be able to tailor the applications to their needs. This will contribute to easier use of XAL at sites where English is not the primary language.

EUROPEAN SPALLATION SOURCE

The European Spallation Neutron Source (ESS) is a project to design and construct a next-generation facility for research with beams of neutrons. At 5 MW beam power, The ESS will be the brightest source of neutrons in the world, enable scientists across many disciplines to perform experiments and investigate materials. The ESS will also retain and strengthen the current European position in the neutron science [6].

The ESS will be composed of a high-current proton linac, which will deliver 5 MW of power to the target at 2.5 GeV, with a nominal current of 50 mA [7]. With respect to the controls conceptual design, the machine will be similar to SNS. ESS has therefore planned to take advantage of experience and expertise developed at SNS, including standardization of a Controls Box environment for distributed R&D and development among partner laboratories [8], and use of the XAL framework as a solution for high-level applications and accelerator physics tool development.

XAL AT ESS

An XAL Workshop was held at SNS in May 2010, where current and potential future users of XAL discussed future goals and framework development. Attendees represented 11 organizations, including ESS, SNS, FRIB, BNL, TRIUMF, and CSNS [9]. Part of the outcome of the workshop was the previously mentioned plan for refactorization of the XAL libraries and structures. This refactorization should encourage development by users other than SNS. It will introduce much more modularity and easier maintenance; cooperation and sharing among users will also be easier.

Open XAL will be split into several different parts, each responsible for a particular group of functionalities (e.g. separating the core from the devices model, fully detaching the database access layer, etc.). Each user of XAL could then decide what modules to include in their distribution, which devices implementations are required by the machine, and so on.

At present, ESS has a partially constructed lattice database. Multi-particle beam dynamics for the linac has been studied using the TraceWin code [10]. The results of these studies and simulations have been entered into the MySQL database. A dedicated Java application has been

written to gather the data from the lattice database and constructs the accelerator model from the Java objects. This model can be serialized to an XML file, using the document definition specified by XAL. All the devices and other data that are required by XAL (e.g.. power supplies for magnets, epics channel names), but are yet missing in the lattice database are filled with dummy data to enable XAL model use at the earliest development stages.

Because there is no available ESS EPICS database yet, the model can only be used in simulation mode. A dedicated application called *Virtual Accelerator* is provided by XAL, which loads a specific accelerator sequence and simulates EPICS channels using the Channel Access Server (CAS) [11]. The values simulated by the CAS are calculated by the online model. Due to the nature of the CAS, the simulated channels can be used as any other EPICS PV and therefore, XAL can also connect and use those channels directly without the need to modify any part of the code.

The virtual accelerator feature will play an important role during the development and commissioning period of the ESS accelerator. It permits tests of features and the design of the accelerator without the need to connect to a fully implemented control real system. Users will be able to develop software without concerns about early integration problems.

The next step in adaptation of XAL for ESS is the addition of new devices and potential adaptation of the existing devices. ESS will use certain types of physical devices that are not used at SNS and are therefore non-existent in XAL. These new devices will have to be implemented and added to the XAL model. The optical properties of the new devices will also have to be implemented to allow use of the devices in the online model to perform beam dynamics simulations.

CONCLUSIONS

XAL has been a collaborative project from the beginning with roots in Brookhaven's Unified Accelerator Libraries, with contributions and interest from other labs around the world [12], though the main effort of the project has been to deliver applications for SNS. This has resulted in fragmentation of the code among various contributors. There has been a recent effort to organize and coordinate the project. XAL's position as a useful accelerator application framework will be strengthened by new laboratories joining the collaboration. The use of a well-developed framework and tested applications will improve early adoption of control application standards, and should ease the commissioning period of the ESS.

REFERENCES

[1] J. Galambos et al, "XAL Application Programming Framework", ICALEPCS 2003 Proceedings
[2] J. Galambos et al, "XAL Application Programming Structure", PAC 2005 Proceedings
[3] J. Galambos et al, "XAL – The SNS Application Programming Infrastructure", EPAC 2004 Proceedings
[4] C. M. Chu et al, "SNS Application Programming Infrastructure and Physics Applications", APAC 2004 Proceedings
[5] T. Pelaia II, "XAL Application Framework and Brick GUI Builder", ICALEPCS 2007 Proceedings
[6] http://www.ess-scandinavia.eu/
[7] M. Eshraqi et al, "Conceptual Design of the ESS LINAC", IPAC 2001 Proceedings
[8] T. Satogata et al, "ESS Controls Strategy and the Control Box Concept", these proceedings.
[9] http://neutrons.ornl.gov/conf/XAL2010
[10] http://irfu.cea.fr/Sacm/en/logiciels/index3.php
[11] http://caj.cosylab.com
[12] T. Pelaia II et al, "XAL Status", ICALEPCS 2007 Proceedings

CCCP - COSYLAB COMMON CONTROL PLATFORM

Miha Rescic, Cosylab, Ljubljana, Slovenia
Ziga Kroflic, University of Ljubljana, Ljubljana, Slovenia

Abstract

Cosylab common control platform (CCCP) is a lightweight hardware control platform designed to provide a simple interface to various types of hardware components and fast and simple integration of such hardware into control systems. The core of the platform is the scripting language lua. This lightweight and flexible scripting language provides software real-time control of hardware modules over all provided connections (RS232, Ethernet, USB, SPI, CAN, I2C, GPIO) as well as fast and simple ways of implementing modules for more complex structures (FPGA). The platform provides various levels of control with an embedded GUI or full remote control over an embedded web server, archiving capabilities with a database back-end and different device simulator modes. The platform's small footprint, high degree of flexibility and high level of hardware abstraction make the CCCP an ideal control platform for more complicated hardware instruments and at the same time a perfect main control board for devices that incorporate various complex hardware elements. The design and possible implementations of this platform will be discussed in this article.

INTRODUCTION

Development of a control system is never an easy nor a straightforward task. With the complexity of today's technologies, if we're speaking of technologies in general or of technologies applied in specific fields, the number of different components or building blocks of the control systems and the complexity overall grow rapidly.

Within this rapidly expanding field it is very difficult to find a common ground and usually much effort is spent on developing highly specific solutions capable of tackling only a limited array of problems. Thinking of common grounds in control systems field brings to mind a reusable, as generic as possible platform that would represent the base of the control system. This was the motivation behind CCCP: minimize the efforts needed for base platform development and allow emphasis on more specific and complex components development, integration, testing and QA.

ARCHITECTURE

The crucial element of the platform is the architecture. CCCP tries to keep logical entities separated from each other as much as possible. This way, reusability and efficient design are possible.

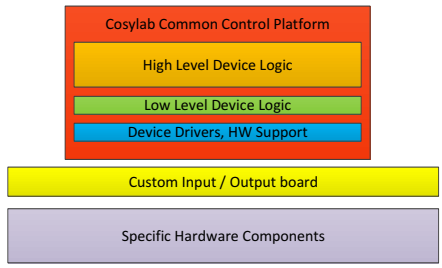


Figure 1: CCCP Architecture

Custom input / output board

On the lowest level of the CCCP architecture is the customized input / output board. Although the board itself is not a part of the CCCP platform it provides problem or component specific solutions regarding hardware connections, specific protocol implementations or more advanced logic (see Fig. 2). The custom board development is bundled together with the CCCP platform development in order to provide the optimum solution for the specific problem.

Some of the IO board's main purposes are described below.

- Target hardware development away from the platform core and towards specific implementation needs.
- Provide advanced logic and (hard) real-time support with FPGA.
- Allow connectivity with existing CCCP IOs or implementation of any custom IO required.
- Minimize the complexity of custom HW development.
- Minimize the amount of redundant development efforts regarding non-reusable hardware.

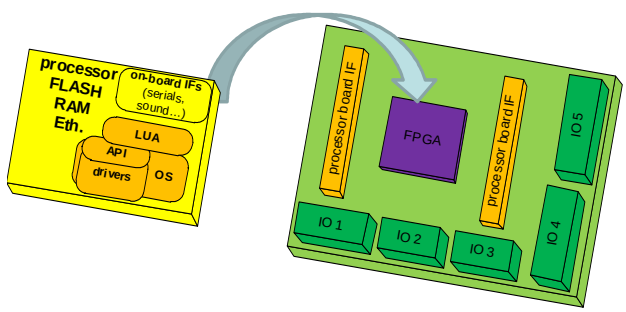


Figure 2: Custom IO board

Device drivers and hardware support

The layer residing directly over the custom IO board, the lowest layer of the CCCP core architecture, provides

all the needed logic to communicate with hardware. It contains two crucial elements:

- OS specific drivers providing basic system input / output functionality.
- HW modules providing an interface layer between the underlying hardware components and higher-level device logic.

The HW modules not only allow high level logic to easily interact with the underlying hardware but create an abstraction layer between the two that can be replaced by a mock layer in absence of actual hardware. A mock layer or a simulated layer makes development possible without actual hardware and allows more flexible testing (without hardware limitations) and much faster integration.

Low level device logic

The low level device logic incorporates all the services and layer logic needed for the CCCP platform to function properly. They lay the foundation for the higher layer logic and provide the tools that allow developers and engineers faster development.

Some of the main components residing in the low level device logic:

- HTTP server for northbound communication and control.
- Priority task scheduler with support for interrupts from HW modules.
- Lightweight database for storing data, events and all-purpose logging.
- Generic FIFO queues for inter-process data exchange.

Most of the low level logic is written in C programming language but some segments also use components written in the scripting language lua.

High level device logic

The highest CCCP architectural layer is where the magic happens. This layer, also called the “instrument” logic layer, is developed entirely with the scripting language lua.

The choice of scripting language over a programming language has at least these advantages:

- All the complex implementations are done in lower layers thus abstracted away from the developer.
- Because of simpler syntax, robustness and user friendliness scripting languages make development available to other team members as well, e.g. engineers.

The use of a higher level of logic together with an application and UI framework (e.g. Nokia’s Qt) makes it possible to further upgrade the device with GUIs and other device interfaces (touch screens, ...).

COMMON CONTROL PLATFORM

In order to provide a truly common platform there are some aspects of the platform that need to taken into consideration.

Customizability

Common platform must provide enough flexibility to allow easy customisation for various implementations. Therefore, the core CCCP has no direct IO connectors or switches. It only provides a standard TX-DIMM connector with standard pinout. In order to connect the common control platform to corresponding control system components a separate IO board must be developed.

By mechanically separating the logical parts into two components (CCCP and the IO board) we achieve a high degree of flexibility and customizability. With the custom IO board approach the solution can be very problem specific but still at the same time very generic since all of the core logic is kept on the CCCP platform. The IO board merely serves as an interface to hardware components whereas the implementation of the logic resides on the generic CCCP board and can be further reused in other various control systems or subsystems.

Size and form factor

One of the first limitations a standard common platform encounters is its size and form factor (see Fig .3) but the size of CCCP (DIMM200-module standard size: 67.6 mm x 26 mm x 3.6 mm) makes is suitable for almost any application.

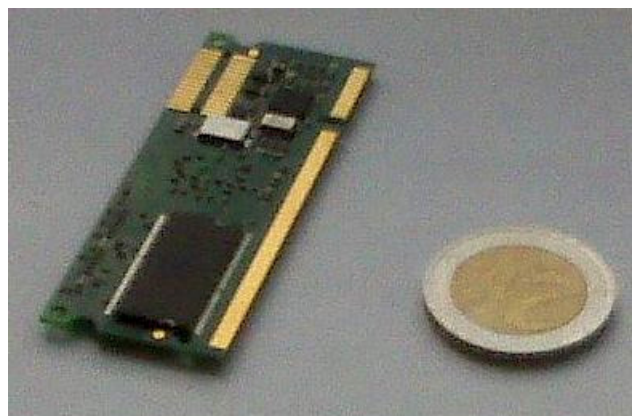


Figure 3: CCCP size

Processor and operating system

The other important aspect of the common platform is the choice of the processor and the operating system. This is why CCCP is powered by an ARM9 400 MHz processor with the operating system of choice being Linux running the 2.6 kernel.

The combination of ARM processor and Linux OS allow users and developers to use a wide range of existing tools, from cross-compilers to integrated development environments.

Connectivity

In order to connect various components to the common platform a number of standard IOs must be supported. CCCP provides the following possibilities (only the most common options are mentioned):

- 10 / 100 Ethernet
- 5 x UART
- 3 x SPI
- 3 x I^2C
- 4 x GPIO
- 3 x 12 bit ADC
- 2 x CAN

PRODUCT COMPONENT SIMULATION

The most important aspect of CCCP is the possibility to substitute real hardware components with mock or simulated components (see Fig. 4).

The architectural layering described above, especially its *Device driver and HW support* layer allows a smooth interchangeability between real hardware components and software-simulated components. Because the hardware modules are essentially exposed to higher level device logic it is, after all the interfaces have been defined and with the use of lua flexibility, quite straightforward to make the switch. The simulated device components are implemented at a higher level of logic (in the high-level, lua logic layer) therefore they are overriding any actual hardware components.

Agile development

The process of mocking or simulating absent hardware components makes it possible to introduce new approaches to otherwise rigid hardware development field. One of these approaches is agile development. Some of the benefits:

- Difficult and complex tasks can be dealt with earlier.
- Problems and complications are discovered earlier and therefore resolved earlier.
- Development process can be split into multiple tasks from the beginning and therefore modified based on completion of and feedback from such tasks.

Test driven development

Testing in hardware development is usually the last stage of development process. With the introduction of simulated components the testing can take place from a very early stage onwards.

- Every step of development can be backed up and controlled by matching tests.
- Tests provide feedback and allow the agile process mentioned above to function properly.

REAL-WORLD IMPLEMENTATIONS

Some of the possible use cases of CCCP control platform are described below.

Remote hardware control

One of the basic examples of CCCP usage would be remote monitoring and control of hardware devices, e.g. household appliances.

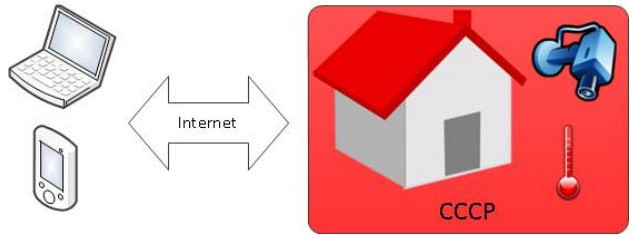


Figure 4: Household control and monitoring

Specific instrument interface

CCCP could also provide an interface to various complex instruments and simplify the integration of these components into the control system.

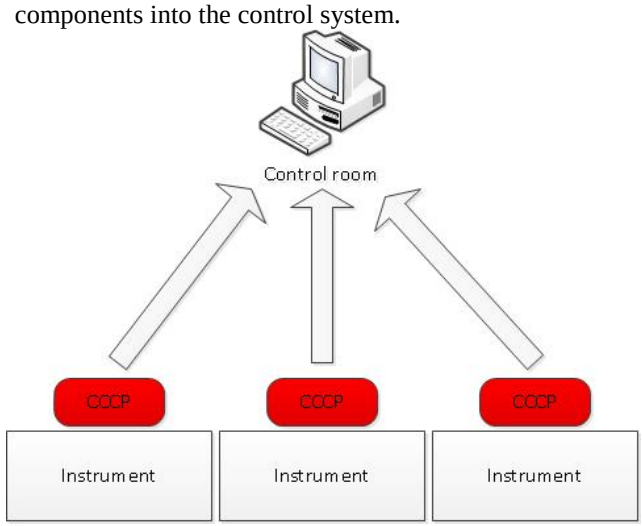


Figure 5: Specific instrument interface

CONCLUSION

Cosylab Common Control Platform presents a different approach to a somehow rigid field of hardware development. With the modular approach regarding hardware and software architecture, simple input and output interfaces, flexible scripting language core logic and device component simulation capabilities it gives our customers a number of benefits.

- Faster time to market with lower development costs.
- Better developer utilization and efficiency. Faster hardware integration, validation and verification.
- Minimized overdevelopment and complexity with maximized flexibility.
- Optimized development process by test driven development and task segmentation.

Small footprint, high degree of flexibility and high level of hardware abstraction make the CCCP an ideal control platform for complicated hardware instruments.

PROGRAMMING INTERFACES FOR RECONFIGURABLE INSTRUMENTS

Matej Kenda, Hinko Kočevar, Tomaž Beltram, Aleš Bardorfer, Instrumentation Technologies d.d, Solkan, Slovenia

Abstract

Application Programming Interfaces (APIs) provided by the manufacturers of the instruments for the accelerators are a very important part of the functionality. There are many interface standards (EPICS, TINE, Tango,...) and even same standard can be used in various ways.

Important features of modern instruments are reconfigurability and embedded computing.

The developers of instruments that need to be connected to a control system are facing different requirements: adherence to standard protocols and support of reconfigurable instruments with diverse capabilities with a consistent interface.

Instrumentation Technologies has implemented a well accepted solution with its proprietary Control System Programming Interface (CSPI) layer and adapters for each standard protocol.

There are new challenges like reconfigurability, quality of service, discovery and maintainability that are being addressed with improved Measurement and Control Interface (MCI).

CONTROL SYSTEM AND SOFTWARE INTERFACES

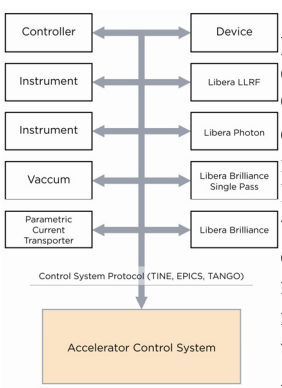


Figure: 1

There are quite some parameters that define environment in which the Control System operates. We can find heterogeneous instruments with different levels of complexity. Beside that the equipment is distributed over large remote regions and needs to provide reliable access regardless of the distance from the control room (see Fig. 1). Another characteristic of such operating environment is that the control is centralized, but the data acquisitions is distributed and to some extent also the data processing.

Based on that we can define interface requirements from the Control System's point that must cover following areas:

- device discovery, identification and capabilities
- operation mode control and configuration parameters
- events, alarms and health state monitoring
- data acquisition and attributes (data type, size, offset, time-stamp)
- error handling

INSTRUMENT MANUFACTURER'S VIEW

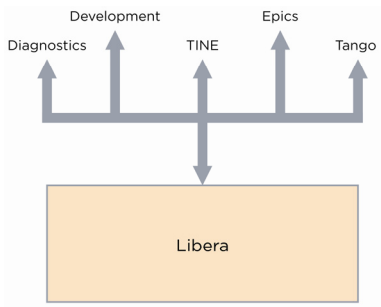


Figure: 2

From the reverse point of view, an instrument can be used in different environments (see Fig. 2). Requests for data can come from different sources for different purposes.

- **Control System**: Different types of control system protocols
- **Other instruments**: Instrument interoperability, multiple instruments working together, clustering, shared processing,
- **Development Lab**: Development, testing of new, updated instruments
- **Maintenance**: Diagnostics, repair

Not all of the access paths are active concurrently.

A great deal of the information access has a common denominator, defined by the type of the information requested.

EMBEDDED COMPUTING

Using embedded computers in the instruments enables instruments to behave as network attached devices with built-in control system interfaces.

Embedded computer can be used to

- **control** the instrument's operation
- perform a part of **digital signal processing**
- provide **remote access** to the instrument

The embedded computer is one of the important components of an instrument, because it provides convenient way to bring all of the parts (hardware modules, FPGA, software) of an instrument together into a working application and perform certain digital signal processing.

Software running on the embedded computer can seen as one of the variable parts of a reconfigurable instrument.

RECONFIGURABLE INSTRUMENTS

Physical setup and behaviour of the instrument is not completely defined during manufacturing.

Modern trends in development of instrumentation encourage modularity with many standards for physical dimensions, electrical interconnectivity and data exchange protocols (see Fig. 3).

This leads to the following combinations:

- **Reuse of modules**: Hardware module MOD_A can be used in instrument INS_A, INS_B, …
- **Behaviour** of the hardware module MOD_A can be altered by loading different FPGA designs
- Instrument INS_A can comprise **variable number of modules** MOD_A, MOD_B, MOD_C, thus defining different **variations of the instrument**.

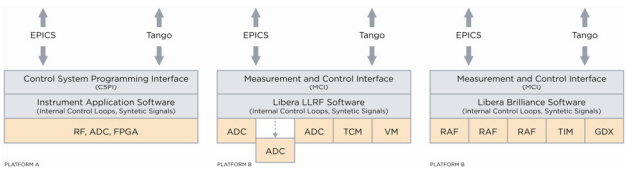


Figure 3

Design of the software, running on such an instrument, must be done in a way to recognise and make use of these combinations.

In general, the responsibilities of the instrument software can be split in several semi-independent layers: managing hardware platform, instrument application logic, external interfaces.

Hardware flexibility influences all of the software layers, including external interfaces.

Semantic Types of Information

The information transferred between the CS and the instrument can be divided into: digital signal acquisition, alarms (notifications), monitoring and control of the instrument state and behaviour (see Fig. 4).

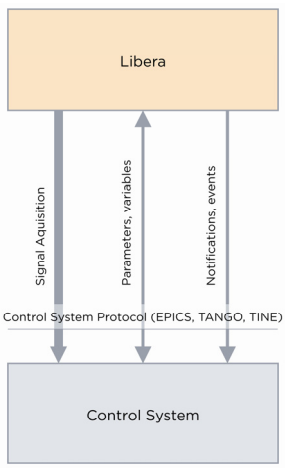


Figure: 4

Time considerations in the data transfers involves data rate and frequency. That is the time that is needed to transfer certain amount of data and the repetition speed how often that transfer happens.

Every data has its origin (data provider, source) and its destination (data consumer, sink). Depending on the active or passive involvement of either side in the data flow we can distinguish between data stream (data provider push) or data on demand (data consumer pull) as depicted in Table 1.

Table:1

	History Buffer	Streaming
Data access	Pulled by user (on demand)	Pushed by instrument (on trigger)
Size	Large	Small
Example	Turn by turn, ADC	Slow acquisition, fast acquisition, events

PROGRAMMING INTERFACES OF LIBERA INSTRUMENTS

Instrumentation Technologies develops families of specialised instruments for use in the accelerators. They are all equipped with embedded computers and have network connectivity.

Instruments can be divided in two classes: **Platform A**, **Platform B**. Main difference in hardware is the level of modularity, reconfigurability and computing power.

Modern trends in instrumentation required Libera instruments to evolve and become more modular and reconfigurable. Platform B instruments comply to μTCA, IPMI and other standards and comprise powerful embedded computer. Software, developed for these instruments had to be modified as well to support and utilise new hardware platform.

The goal of programming interfaces on both platforms is similar: implementation of as much functionality as possible in a common fashion and converting that information to a specific control system protocol as late as possible.

Both types of interfaces provide access to the semantic types of information described above (see Table 2).

Table: 2

	CSPI	MCI
Networked API	Yes (Generic Server)	Yes
Number of signals	Fixed number	Dynamic (based on setup and processing)
Processing	Control loops	Control loops, more DSP algorithms (depends on application)
Configuration	Numeric identifiers	Registry (dynamic structure)
Notifications	Callbacks	Registry (built-in functionality)
CLI	libera	Multiple
Control system interface	EPICS driver (two versions), Tango driver (3rd party)	MCI to EPICS adapter, more planned
GUI	EPICS EDM, Matlab, custom as implemented by CS integrators	Qt GUI, EPICS EDM

Control System Programming Interface (CSPI)

CSPI is available on **Platform A** type of instruments (Libera Briliance, Libera Brilliance Single Pass, Libera Photon, Libera BunchByBunch). These instruments contain energy efficient ARM based embedded computer with limited computing power.

The operating system, used on the computer, is stripped-down distribution of Debian Linux, running on Linux kernel 2.6.20.

The computer is designated for proper operation of the hardware and FPGA from powering the box on to shutting it down and to provide network connectivity.

Hardware configuration of Platform A instruments is **defined at manufacturing**. Available data and the API are coupled together.

CSPI provides interfaces for:

- **Monitoring, controlling the instrument** through a number of parameters. They are all integer numbers and identified by numeric Ids. The set of parameters is fixed for a certain instrument.
- **Acquisition of the signals**. Functions to easily access pre-defined number of signals are available.

- **Change notifications**. A callback function can be registered, which is called with the ID of the parameter that was modified.

Remote access is provided by:

- **Generic server**: transparent CSPI API access over the TCP/IP.
- Embedded **EPICS driver**: EPICS IOC driver that utilizes CSPI API. Alternative implementation was developed at Diamond Light Source that by-passes CSPI and communicates with the hardware in more direct fashion.
- Embedded **Tango Server**, developed by Elettra institute.
- External **Tango Server**, developed as a collaborative effort between Alba, Desy, Elettra, ESRF and Soleil institutes.
- External **TINE Server**, developed by Desy institute.

Measurement and Control Interface (MCI)

MCI is the interface of the Platform B instruments (Libera LLRF, Libera Brilliance+, Libera Single Pass H).

Platform B instruments contain various types of i386-based embedded computers. These computers run standard Ubuntu Server edition (Linux kernel 2.6.26 or 2.6.32).

Dynamic nature of Platform B instrument required different design approach of the software and its API.

MCI has separated classes and functions of the API from the information that they are used to access. MCI is networked by design.

The following concepts have been introduced in the API:

- Registry: tree-structured representation of information, used to monitor and control parameters of an instrument.
  - The tree nodes are **populated by the instrument software dynamically**, depending on the hardware setup and type of the instrument
  - Nodes can emit **notifications** (for example: value change). Callbacks functions can be registered to nodes to receive those notifications
- Data Streams

Remote access is provided by

- Directly by **MCI**
- **EPICS adapter**: lightweight server without a database maps MCI registry and signals to EPICS PVs
- **Tango, Tine adapter**: will be developed when needed

Examples

Sample command line tool for reading the Libera unit environment parameters with CSPI

```
$ net-libera -i 10.0.0.100 -l
          Temp [C]: 45
         Fans [rpm]: 4590 4560
     Voltages [mV]: 1489 1782 2439 3233 4892 11865 -12020 -5089
```

Example of source code:

```
// Connect to the Libera unit at IP address  10.0.0.100
server_connect ("10.0.0.100", 23271, "224.0.1.240", 0);
// Allocate the environment handle
cspi_allochandle (CSPI_HANDLE_ENV, 0, henv);
// Prepare variables for environment parameter readout
CSPI_ENVPARAMS params;
CSPI_BITMASK mask = ~(0LL);
// Acquire the parameter
cspi_getenvparam (henv, &params, mask);
// Release the envirnment handle
cspi_freehandle (CSPI_HANDLE_ENV, henv);
// Disconnect from the Libera unit
server_disconnect ();
```

Structure of MCI registry as presented by a sample command line tool.

```
$ ./libera-ireg dump -h 10.0.3.40 -l 3
IP_10-0-3-40
  boards
    raf5
    chassis:0
    chassis:1
    chassis:2
    chassis:5
    os
$ ./libera-ireg dump -h 10.0.3.40 -l 3
boards.chassis:1.board_info
board_info
  type = VM
  status = Running
  power_status = Mng + Main
  fpga_revision = 7103
  ipmi_version = 81
```

Example of source code:

```
Using namespace mci;
// Connect to instrument 1
RemoteNode h1 = CreateRemoteRootNode("10.0.33.1", 5678,
"libera-platformd");
Node r1(h1);
// Connect to instrument 2
RemoteNode h2 = CreateRemoteRootNode("10.0.33.2", 5678,
"libera-platformd");
Node  r2(h2);
// Query specific temperature from ins 1
Node tempNode = r1.GetNode( {"boards", "chassis:0", "sensors",
"ID_2" } );
float temp = tempNode.GetValue();
```

REFERENCES

[1] CSPI Reference Guide; Instrumentation Technologies d.d.; http://www.i-tech.si

[2] MCI Reference Guide; Instrumentation Technologies d.d.; http://www.i-tech.si

[3] Experimental Physics and Industrial Control System (EPICS); http://www.aps.anl.gov/epics/

[4] TAco Next Generation Objects (TANGO); http://www.tango-controls.org/

[5] Intelligent Platform Management Interface; http://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface

[6] MicroTCA; http://www.picmg.org/v2internal/resourcepage2.cfm?id=5

[7] Gstreamer; http://www.gstreamer.net/

EPICS IOCCORE REAL TIME PERFORMANCE MEASUREMENTS ON COLDFIRE MODULE*

Shifu Xu#, Hairong Shang, Robert Laird, and Frank Lenkszus
Argonne National Laboratory, Argonne, IL 60439, U.S.A.

Abstract

Since Experimental Physics and Industrial Control System (EPICS) is becoming more widely used in accelerator control systems and the EPICS Input/Output Controller (IOC) has ported to different operating systems, the performance of EPICS IOCcore on different hardware and software platforms is crucial. This paper will provide real-time performance measurements of EPICS IOCcore on a Coldfire module uC5282 and on two different OS platforms: RTEMS 4.9.2 and uClinux 2.6.21. The most recent EPICS base and extensions are used to build the test application.

INTRODUCTION

As more and more Coldfire uC5282 modules are being used at the Advanced Photon Source (APS) and other sites, it is of interest to know the EPICS IOCcore real-time performance on this platform. Similar performance measurements were done on the MVME2100 [1]. Based on the measurement software [2], a few changes have been made to measure on the Coldfire uC5282 module. These real-time parameters are measured on both RTEMS 4.9.2 and uClinux 2.6.21 platforms: interrupt latency, context switch latency, and total response latency. Two more parameters are measured on the uClinux 2.6.21: interrupt top half to bottom half, and interrupt bottom half to user space interrupt service routine (ISR).

MEASUREMENT PLATFORM

All measurements were performed on a Coldfire uC5282 module from Arcturus Networks [3]. The module has a MCF5282 Freescale Coldfire microprocessor with a 64-MHz Coldfire RISC core. It has a 16-Megabyte SDRAM, 4-Megabyte flash memory, and 512-k byte on-chip flash. In order to generate an external interrupt for the module to measure the latency, an APS custom-made Coldfire bridge board and Altera Stratix II development board were used. Figure 1 shows the hardware platform.

The development host machine is an x86-based Linux PC running Fedora Core 10, with a tftp client and an NFS server running on it. The target module's bootloader has a tftp server to receive the OS image.

Two OSs are evaluated on the Coldfire module target: RTEMS 4.9.2 and uClinux 2.6.21. uClinux 2.6.21was downloaded from Arcturus Networks with the non-preemptive kernel. This version includes built-in board support packages (BSPs) for the Coldfire modules. The cross-compiler tools for the uClinux 2.6.21 and applications were also provided by Arcturus Networks. Because of the resource limitations of the Coldfire uC5282 module, efforts were made to optimize the uClinux kernel in order to get better performance.

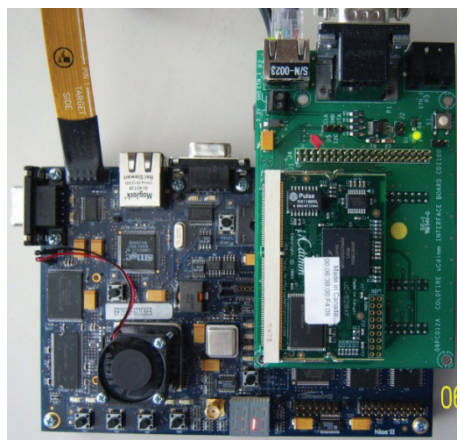


Figure 1: The hardware platform.

The most recent EPICS base 3.14.11 was used for the test. A few new EPICS base configuration files were created for the Coldfire uC5282 module on the uClinux platform.

MEASUREMENT SOFTWARE

The software from [2] is generic EPICS IOCcore performance measurement software for target OSs such as vxWorks, Linux, and RTEMS. Figure 2 shows the software structure.

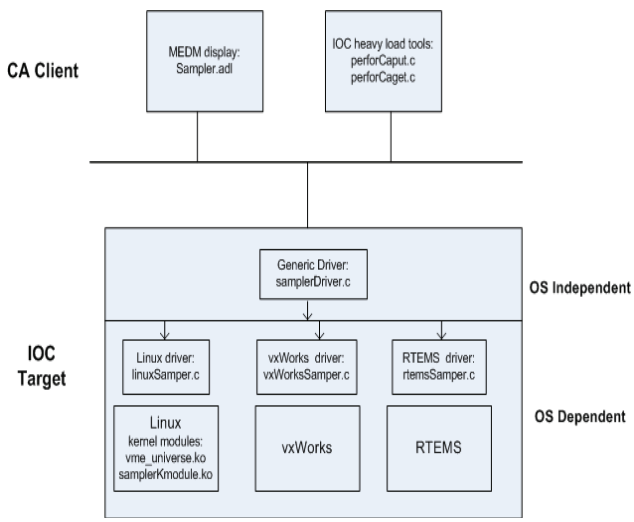


Figure 2: The measurement software structure.

* Work supported by U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.
#xusf@aps.anl.gov

For each target platform, a specific interrupt generation method is needed for the interrupt latency measurement. An external interrupter, which includes an APS custom-made Coldfire bridge board and Altera Stratix II development board, was used for the latency measurement on the Coldfire uC5282 module. A parallel I/O (PIO) component was used as an Avalon slave in the Altera FPGA design to generate interrupts to the Coldfire module. The interrupt generation code resides in the RTEMS-dependent driver rtemsSampler.c. For uClinux, a kernel module was created, which has an interface function to generate this interrupt. A Linux-dependent driver, linuxSampler.c, in the user-space is used to call this interface function.

Due to the limited memory resource on the Coldfire uC5282 module, only 1000 EPICS records were loaded. There are two Channel Access clients that put a load on the IOC: performCaget and performCaput. The performCaput puts values to the records on the IOC. The performCaget monitors the value changes.

USER INTERFACE

A MEDM display was created for operation and showing measurement results. It can configure the number of samples to take with each scan. It can display the minimum, median, maximum, and percentage of samples over some value for each latency parameters. Figure 3 shows the user interface.

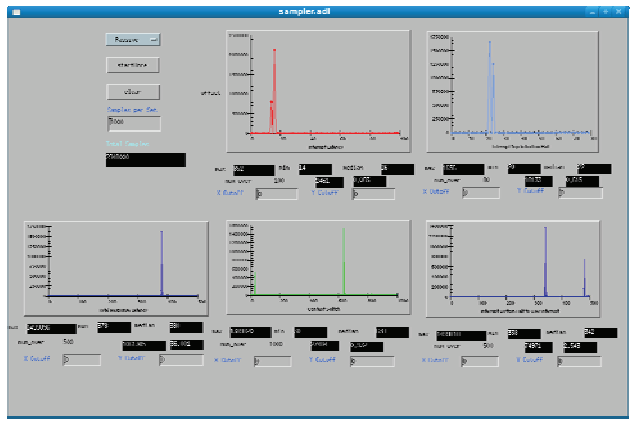


Figure 3: The user interface.

MEASUREMENT RESULTS

The IOC is heavily loaded in all the tests. Four different values of each parameter are collected: minimum, median, maximum, and percentage of samples over some value. Tests on the private network were conducted for one hour. To look for network interference, some tests were run for two hours on a public network. Another test was run to measure user-level interrupt latency. Tables 1–5 show the results. All the units are in units of μs.

Table 1: Interrupt Latency

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	uClinux non-preemptive	12	14	1822	0.05
	uClinux non-preemptive with user level ISR	14	16	852	0.083
	RTEMS net task has higher priority	18	19	142	0.006
	RTEMS net task has lower priority	18	19	131	0.008
Public Network	uClinux non-preemptive	14	14	1926	0.056
	uClinux non-preemptive with user level ISR	14	16	1604	0.101
	RTEMS net task has higher priority	18	19	165	0.006
	RTEMS net task has lower priority	18	19	132	0.006

Table 2: Interrupt Top Half to Bottom Half Latency

OS		Minimum	Median	Maximum	>100 μs(%)
Private Network	uClinux non-preemptive	20	22	1934	0.144
	uClinux non-preemptive with user level ISR	20	22	1656	0.615
Public Network	uClinux non-preemptive	20	22	1932	0.125
	uClinux non-preemptive with user level ISR	20	22	1828	0.605

Table 3: Interrupt Bottom Half to User Level Interrupt Latency

OS		Minimum	Median	Maximum	>500 µs(%)
Private Network	uClinux non-preemptive with user level ISR	338	342	1499818	2.543
Public Network	uClinux non-preemptive with user level ISR	338	342	1264560	2.703

Table 4: Context Switch Latency

	OS	Minimum	Median	Maximum	>100 µs(%)
Private Network	uClinux non-preemptive	28	30	121464	0.482
	uClinux non-preemptive with user level ISR	30	638	1389820	0.932*
	RTEMS net task has higher priority	44	46	1934	0.077
	RTEMS net task has lower priority	44	46	158	0.032
Public Network	uClinux non-preemptive	28	30	113374	0.481
	uClinux non-preemptive with user level ISR	30	638	1440814	0.914*
	RTEMS net task has higher priority	44	46	2013	0.152
	RTEMS net task has lower priority	44	46	161	0.056

* over 1000 µs(%)

Table 5: Total Response Latency

	OS	Minimum	Median	Maximum	>100 µs(%)
Private Network	uClinux non-preemptive	80	84	121518	0.81
	uClinux non-preemptive with user level ISR	378	380	1499856	36.001**
	RTEMS net task has higher priority	63	65	1954	0.19
	RTEMS net task has lower priority	63	65	177	0.229
Public Network	uClinux non-preemptive	80	84	113580	0.799
	uClinux non-preemptive with user level ISR	378	380	1264638	37.531**
	RTEMS net task has higher priority	63	65	2033	0.264
	RTEMS net task has lower priority	63	65	181	0.171

** over 500 µs(%)

MEASUREMENT RESULTS ANALYSIS

The results show that RTEMS has better real-time performance than uClinux. Compared with the real-time performance results on MVME2100 [2], it seems that the MVME2100 has better performance than the Coldfire uC5282 module, though the RTEMS and Linux versions are different. Measurement on the uClinux with a preemptive kernel should be conducted in the future for further comparison.

REFERENCES

[1] S. Xu and M. Kraimer, "Real-Time Performance Measurements of EPICS iocCore," ICALEPCS'2005, Geneva, Switzerland, October 2005, PO2.075-5 (2005); http://www.JACoW.org.

[2] http://www.aps.anl.gov/epics/modules/soft/realTimePerform/index.html.

[3] http://www.arcturusnetworks.com/products/uc5282.

HIGH LEVEL MATLAB APPLICATION PROGRAMS FOR SPEAR3[§]

J. Corbett *et al*, SLAC, Stanford, CA 94309

Abstract

The SPEAR3 control system nominally operates with the EPICS toolbox on top of VMS hardware. The simultaneous use of Matlab Middlelayer (MML) and Accelerator Toolbox (AT) allow for parallel, high-level machine control and accelerator physics applications that communicate with the control system via EPICS Channel Access (LabCA). While the majority of the MML and AT software is machine independent, site-specific high-level applications are also required to control the accelerator. This paper describes several such high-level application programs that have been developed for control and diagnostics at SPEAR3. Examples include a time-dependent waveform display gui, beam steering applications, transport line optics correction, SR beam diagnostics and add-ons to the main MML routines.

INTRODUCTION

The SPEAR3 light source came as the result of a Basic Energy Sciences committee recommendation following a review of U.S. synchrotron radiation facilities in 1997 [1]. Before formal DOE/NIH funding arrived in 1999, preliminary lattice design and system engineering specifications were developed on project seed money. During this time, it became clear that the historical, yet dated, SPEAR control system would need to be largely replaced [2], in particular the high-level application programs. The new system would utilize EPICS operating on a VMS platform which opened up the possibility for Channel Access communication with external programs. In order to consider options for modern application development platforms, a satellite meeting was arranged at the 1998 International Computational Accelerator Physics Conference in Monterey, CA. Presentations included options for SDDS, TCL/TK and X-Windows software.

At the time of the Monterey conference, Matlab was already in use at SSRL for data processing and off-line accelerator physics calculations. Matlab had also been used extensively at the SLC for data acquisition, data reduction and to some degree machine control. At the ALS in Berkeley, Matlab was in use for command-line driven machine control and data processing [3], and had the interesting feature that the top-level language closely mimicked accelerator simulation programs such as TRACY [4]. At the same time the first versions of the Matlab Accelerator Toolbox [5] utilizing TRACY transport physics were available for simulation studies at SSRL.

During the Monterey meeting, a proponent of IDL made an interesting observation – since recent versions of Matlab contained graphical interface commands why not use it to develop high-level application programs [6]? With Channel Access connectivity embedded in Matlab (LabCA) [7], a complete solution was available with control system communication, gui capability, user-friendly data reduction software and accelerator simulation tools that could be integrated into a single, all-in-one software package. The gavel fell and a new project was born – high level application programs at SPEAR3 would be developed and written in Matlab[†].

In a stroke of luck, the main author of Matlab Middle Layer (MML) [8] was finishing work on an SBIR grant at SLAC and was available to consult with SSRL on application development for SPEAR3. The first project was to convert the FORTRAN version of the Linear-Optics-Closed-Orbit (LOCO) program to Matlab [9]. It was then recognized that SPEAR3 needed a 'middle layer' to provide easy connectivity between the accelerator physicist and storage ring. By introducing Matlab code utilizing accelerator modeling syntax developed at the ALS, a straight-forward database-drive system was devised for simulation and control.

As more of the ALS software was integrated into the system, the functionality of higher-level programs such as, orbit, tune, dispersion and chromaticity measurement expanded. In order to retain the ability to pass the new software back to the ALS, programs were written in a 'machine independent' format driven by simple MML initialization files to associate accelerator elements and their indices with girder locations, database channel names, hardware limits, conversion factors and specific locations within the AT lattice file.

First tests of machine independence were made in trials at the Canadian Light Source and then again at the ALS. Interestingly, machine-independence also created a structural rigor within the software that ultimately simplified high-level program development and streamlined switching between on-line and simulation control modes. Hardware-to-physics conversion factors also enabled the user to 'switch' between hardware (e.g. amps) and physics (e.g. m^{-2}) units with a single command. Similarly, the AT lattice pointers automate switching between on-line and simulation modes with a single command. File directory specifications were then incorporated to automate data file look-up and data storage needs for machine control and simulation.

In the sections to follow we describe high-level application program developments at SPEAR3 in the areas of waveform variable display, main ring and transport line machine tuning and optical diagnostics.

[†]the philosophy was, and still is, 'anything that can be written in EPICS will be written in EPICS'.

[§]Work supported by US DOE Contract DE-AC03-76SF00515 and Office of Basic Energy Sciences, Division of Chemical Sciences.

CONTROL/DISPLAY APPLICATIONS

One of the first high-level MML application programs was a real-time orbit monitor program intended to compliment a manual orbit control program. It was soon recognized that the x- and y 'orbit' families were only two instances of a larger class of 'families' defined in the MML initialization files. By simply redefining the 'family' and the associated axis coordinates within the display, a broad selection of accelerator hardware families could be displayed in the same graphical interface. 'PlotOrbit' was converted into 'PlotFamily' including a complete set of options to display data in terms of absolute or relative values, and with interactive axis scaling features. By utilizing the built-in functionality available in the MML, 'saved' and 'golden' family data could be easily recalled into the graphical display.

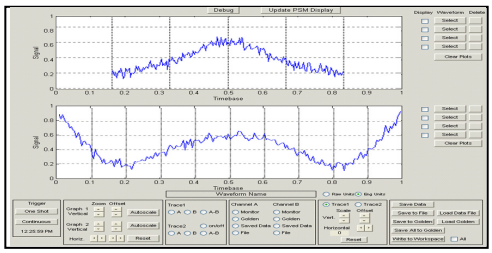


Figure 1: PlotWaveform graphical interface.

To further expand the PlotFamily interface, callbacks from main pull-down menus at the top of the display were programmed to execute other high-level MML code. This feature gave machine operators the capability to load and save entire machine configurations, measure and save machine parameters (dispersion, tune, chromaticity, LOCO data), and a means to control global accelerator properties (orbit, tune, etc). Graphical data could also be exported to the main Matlab workspace for further processing. The resulting PlotFamily application was machine-independent and could operate at any accelerator configured to run MML.

As an extension of the built-in functionality, PlotFamily has an added file execution option that executes at run time. This feature is used at SSRL to generate SPEAR3-specific menu options for transport line control, orbit-interlock checks, machine-specific diagnostic controls and links to hardware documentation.

A further development undertaken at SSRL was to incorporate the PlotFamily display features into a new, high-level 'PlotWaveform' graphical interface. As shown in Fig. 1, PlotWaveform provides a means to display real-time EPICS 'waveform' variables. Most of the ~50 EPICS waveform variables at SPEAR3 are supplied by the Pulse Signal Monitor (PSM) system which consists of a distributed set of analog signal amplifiers and digitizer boards to monitor pulsed RF data (few μs), fast-kicker data (few μs) and booster ramp signals (few ms). Similar to the MML initialization concept, PlotWaveform is based on a machine-specific initialization file that identifies common waveform names with Channel Access names, physical units and time base parameters.

MACHINE CONTROL APPLICATIONS

An early Matlab application program developed for SPEAR3 was the SVD-based orbit control interface 'OrbitGUI' [10]. The control interface utilized Matlab graphic features such as select-and-drag for beam position monitor icons while the underlying software utilized the MML library to open the corrector-to-bpm response matrix file, measure the beam orbit and load both RF frequency and corrector setpoints. The OrbitGUI program was nearly machine-independent with local specifics related to the fact that the code pre-dates MML.

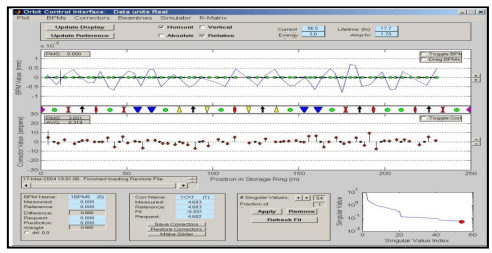


Figure 2. OrbitGUI graphical interface.

The main processing algorithm within OrbitGUI was then transferred to a slow orbit-correction feedback application (SOFB), which uses a Matlab timer object as the internal clock. The SOFB interface is more compact than the OrbitGUI interface allowing only timer on/off and RF correction on/off control. The internal SOFB orbit correction algorithm was updated to allow eigenvector-by-eigenvector mode discrimination. In this case, at each correction cycle SOFB calculates the inner product between the orbit vector and each orbit basis vector in the linear algebra sense. The correction is only applied if the inner product exceeds a pre-specified threshold for each mode. Operationally the discrimination algorithm better rejects BPM noise and results in a quieter beam orbit at the user beam lines.

A similar interactive orbit control program was developed for the linac-to-booster transfer line (LTB). In this case the response matrix is for 'open' as opposed to 'closed' beam orbits and the BPM data requires averaging for accurate results. In order to reliably steer the beam through the LTB, the initial launch conditions (x,x',y,y',dp) must be measured and held constant to minimize mis-steering and dispersion generated upstream. LTBOrbitGUI and the associated response matrix measurement software utilize MML commands are fully integrated into the MML file directory system.

For the booster-to-storage ring (BTS) transfer line, a more complex software system was developed to calibrate the beam line quadrupole optics using LOCO-style response matrix calculations [11]. The BTS software also contains a steering package designed to optimize beam injection efficiency into SPEAR3 .

The RF bucket select software was originally implemented in Matlab but then converted into an EPICS control panel. The conversion was consistent with the philosophy that straight-forward machine-critical software should be written in EPICS where possible.

DIAGNOSTIC APPLICATIONS

The Matlab middle layer and Channel Access connectivity are also utilized for SPEAR3 optical diagnostics. The x-ray pinhole camera, for instance, acquires the beam image with a PointGrey CCD Flea camera [12] routed through IEEE-1394b Firewire to a standard PC. A software link to the PointGrey camera control library maps the image into Matlab memory for processing and display [13]. The Accelerator Toolbox is used to compute relevant betafunctions at the x-ray beam source point. In the nominal beam monitoring mode the measured beam parameters are written to EPICS using LabCA. During periods of machine development, MML scripts are used to manipulate electron beam position, coupling and emittance as measured by the pinhole camera. A Matlab script developed at the CLS calculates spectrally-integrated Fresnel diffraction integrals to characterize beam propagation from source to screen [14].

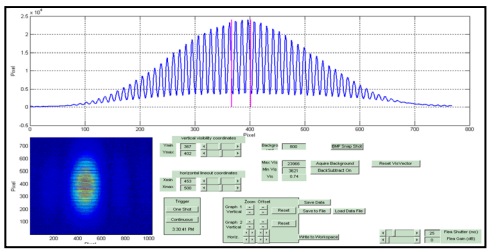


Figure 3: PlotWaveform graphical interface.

A similar, more sophisticated application program was developed for the visible-light interferometer [15]. As shown in Figure 3, a second Matlab-linked Flea camera acquires the raw, 2-slit interference pattern and a graphical interface is used to establish user-defined boundaries for the line-out. A Levenberg–Marquardt numerical fitting algorithm written in Matlab [16] applies a least-squares fit of a sinc/sine function to the interference data to extract the incoherent beam visibility. MML is again used to control insertion device parameters, x-y coupling and emittance for machine characterization.

For the fast-gated and streak cameras, direct links are not available to the internal camera software so raw camera images are saved to disk and re-opened in Matlab for processing. Moments of the transverse and longitudinal beam distribution are fitted to extract data relevant to emittance, machine impedance and instability thresholds. In cases where time-dependent phenomena are recorded, images are pre-processed and then sequenced together in Matlab to generate 'movies' that display non-linear features of the beam distribution that are otherwise difficult to characterize with scalar quantities.

SUMMARY

The Matlab middle layer has provided a relatively user-friendly software package for machine commissioning, operation and accelerator development. Key components include the Accelerator Toolbox, Channel Access Toolbox and a wide range of accessory tools. High-level application programs built largely on the MML allow for scripted data acquisition, data processing and graphical display that are difficult to implement using standard accelerator control system software. To date, over a dozen synchrotron light sources have adopted MML and many have gone on to develop high-level application programs. High-level application programs at SPEAR3 include waveform analysis, beam tuning and orbit control and optical diagnostics. Another important feature of MML and high-level application programs is the ability to provide teaching tools for students and interns.

REFERENCES

[1] R. Birgeneau, et al, 'BESAC Advisory Committee Panel on D.O.E. Synchrotron Radiation Sources and Science', November 1997.
[2] H. Rarback. et al, 'Old Wine in New Bottles-The SPEAR Control System Upgrade', ICALEPCS'99, October 4-8, Trieste, Italy, 1999.
[3] G. Portmann, 'ALS Storage Ring Setup and Control Using MATLAB', LBL LSAP Note #248, June 1998.
[4] H. Nishimura, 'TRACY, A Tool for Accelerator Design and Analysis', EPAC'88, Rome, Italy, 1988.
[5] A. Terebilo, 'Accelerator Modeling with MATLAB Accelerator Toolbox', PAC'01, May 2002.
[6] Harvey Rarback, private communication.
[7] A. Terebilo, 'Channel Access Toolbox for MATLAB', ICALEPCS'01, San Jose, CA, 2001.
[8] G. Portmann and J. Corbett, 'An Accelerator Control MiddleLayer Using Matlab, PCaPAC'05, Hayama, Japan.
[9] J. Safranek, et al, 'Linear Optic Correction Algorithm in MATLAB', PAC'03, Portland, Oregon, 2003.
[10] J. Corbett, 'Orbit Control Using MATLAB', PAC'01, Chicago, Illinois, 2002,
[11] J. Safranek, et al, 'Optimization of the Booster to SPEAR Transport Line for Top-Off Injection', PAC'09, Vancouver, Canada, 2009.
[12] http://www.ptgrey.com/products/flea2/index.asp
[13] Henrik Loos, LCLS, private communication.
[14] Jack Bergstrom, CLS, private communication.
[15] J. Corbett, et al, 'Interferometer Beam Size Measurements in SPEAR3', PAC'09, Vancouver, Canada, 2009.
[16] Xiaobiao Huang, SSRL, private communication.

A NOVEL APPROACH FOR BEAM COMMISSIONING SOFTWARE USING SERVICE ORIENTED ARCHITECTURE*

G. Shen, BNL, Upton, NY 11973, U.S.A.
P. Chu, J. Wu, SLAC, Menlo Park, CA 94025, U.S.A.

Abstract

A novel software framework is under development, which is for accelerator beam commissioning and operation. It adopts a client/server based architecture to replace the more traditional monolithic high level application approach. A minimum set of commissioning and operational services has been defined such as simulation server service, directory service, magnet service, and bpm service, etc. Most of them have been prototyped. Services can use EPICS pvData as its data container and pvAccess as communication protocol. This paper describes conceptual design and latest progress for some services.

INTRODUCTION

Traditionally, an accelerator application needs to deal with many functions such as connection to various signals, data from physics modelling, data plotting, complicated program flow and error handling. If all such computation is built in a single standalone program, the complexity level of the program may result poor performance, unreliability and code maintenance difficulty. Also, if any application needs a new feature which is not provided by an easy interface, it is hard to implement the feature without major restructure of the existing program.

On the other hand, if heavy computation functions can be distributed as running modules residing on various servers and serving up data via proper service protocol, the Graphical User Interface (GUI) application itself can be a simple thin client receiving the data from the servers. This service oriented architecture (SOA) approach can in general improve both performance and reliability of applications.

In this paper, some preliminary result for simulation or model service, Linac energy management (LEM) service and possible communication protocols such as EPICS pvAccess are reported. Work plan for the SOA is also described.

SERVICE ORIENTED ARCHITECTURE

One can identify some essential services for accelerator operation by surveying the functionalities of existing applications. The granularity of services depends on functionality shared by clients, performance, robustness coding complexity, and maintenance. On one hand, too narrow of a service means many more services in total and could cause maintenance trouble. On the other hand, a single service providing too many functions could reduce its performance and reliability. Figure 1 shows a typical top level SOA diagram with a few services.

Furthermore, services can be distributed to multiple servers with virtual machines technology. A distributed system can avoid one service bringing down others. One can also add a redundant server for any critical services.

Advantages for SOA approach are described in detail below.

Easy Application Development

Coding an application with many functions can be tedious. On the other hand, some functions can be shared by several applications. A well-designed SOA approach can greatly reduce the burden on end developers. Applications can then become "thin" clients without much inline computation. Only simple "get/set" data communication with the service providers will be needed. Coding up a complicated application such as controlling an experiment will require much less time and effort. Yet, all the high quality of supporting functionality is fulfilled because the complication is maintained on the server side. This means that even a program written in scripting language such as Matlab script can still have the same high quality of error handling and message logging without additional coding efforts.

Data Control

Because the services are centralized control, i.e. typically only one particular service instance running at a time. This approach can avoid conflict among multiple clients accessing the same device; for instance, feedback and Linac Energy Management (LEM) program might change the same corrector at the same time but magnet server can shedule the two requests properly.

Better Application Memory Management

For individual applications, SOA can avoid large memory and CPU consumption due to heavy computation and data process. Therefore, it can also reduce the chance of client application program crashing.

Service Swappable

It is not necessary to replace all traditional functions with services overnight. One can implement a service at a time. If an old service is replaced by a new one, the application programming interface (API) should remain the same so the client application can pick up the service seeminglessly. This also means the SOA work is highly scalable depending on the available resources. Furthermore, a new service should go through rigorus test before any client application in production can actually use it.

*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

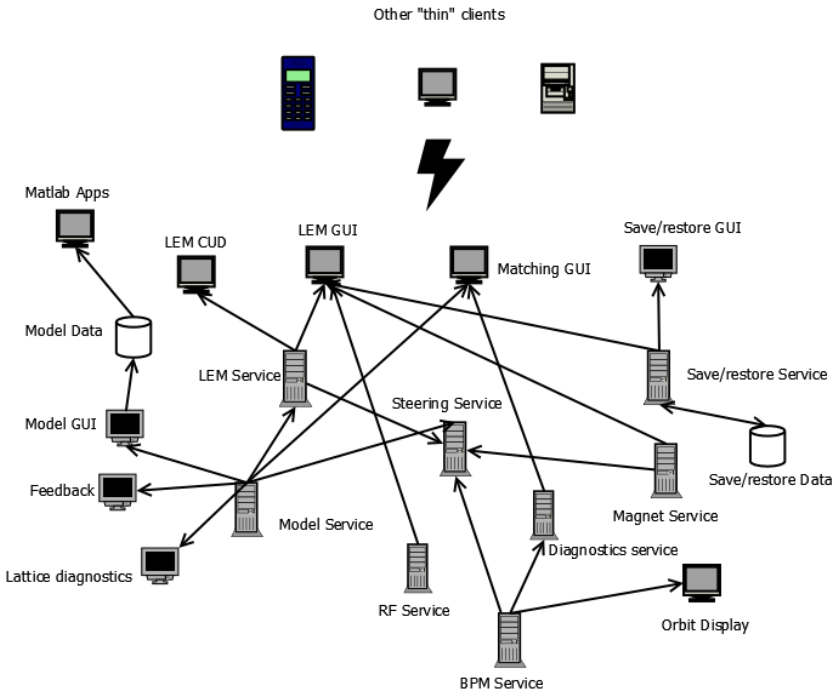


Figure 1: Top level SOA functional diagram. The arrow direction shown in the figure indicates the data flow direction. For instance, Model Service can provide model data to Linac Energy Management (LEM) Service.

SERVICE EXAMPLES

Simulation (Model) Service

Running model within an application is one of the most expensive operations in terms of CPU and memory use. Simulation or model service runs physics model periodically and makes up-to-date model data available for any subscribed clients.

The model server can be expended to cover not only online modelling but also other beam dynamics modelling such as start-to-end simulation, which can provide more detailed beam dynamics simulation information, with a set of uniform APIs. Various simulation codes can be run continuously to supply data to the model server with extant hardware set values.

Figure 2 shows a schematic diagram for the Simulation Service. The core part of the service is a model run control program which manages input data and file preparation, job submission, run status monitoring, run forced quit and output data management.

A prototyped run control program with Fortran based IMPACT-T [1] modelling code using Java and Python has been written and under test. Java part of the program is mainly for data display such as tables and plots while Python is excellent for file I/O and communication with the modelling code and the underneath operating system. The run control program dynamically generates a set of IMPACT-T input files based on user's input via GUI. For each run, a new directory named with the run start time is created and all files are saved under the directory.

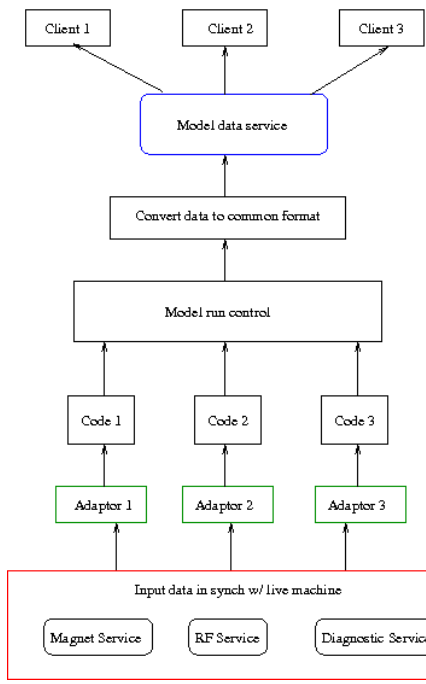


Figure 2: Data flow for model engine and service.

Linac Energy Management (LEM) Service

Any linear accelerator can change its energy from time to time. In order to maintain the same lattice all the time, a program so called LEM which continuously updates the energy information has to run regularly. LEM requires RF data and model tracking; therefore, it is most efficient that it is running periodically on a server and updating all data for clients such as LEM application and control room continuous update display (CUD).

A prototype LEM Service posting calculation result on EPICS Process Variables (PV) has been implemented. Preliminary result shows that the service has been running for over a month even with the accelerator itself being up and down. In contrast, the standalone version of the LEM program crashes easily due to various causes such as data acquisition failure, memory management issue and so on.

Directory Service

This service is prototyped under a sourceforge project so-called epics-pvdata [3,4]. The epics-pvdata consists of 4 modules: (1) pvData, which defines and implements an efficient way to store, access, and transmit memory resident structured data; (2) pvAccess, which is a new generation of EPICS Channel Access protocol. It is used to deliver data over the network and fully supports pvData, and depends only on module pvData; (3) javaIOC, which is a processing engine. All behaviours are defined by JavaIOC engine, and user has only to develop his own support for all desired behaviours. It depends on the pvData and pvAccess; (4) pvService, which is a combination of all services under this project. All generic services or facility specified services should locate here.

The Directory Service, so-called itemFinder, is one particular example under pvService module. It provides a basic function to get a list of physics elements and its associated properties such as EPICS PV names for read-back, set-point, temperature, and so on if they apply. It is designed and prototyped against MySQL relational database (RDB). The RDB schema consists of two (2) tables: (1) *item* table, which stores the physics names for all elements installed in a facility; (2) *property* table, which stores all properties associated with each element.

A client application gives search criteria by calling a client API. The search command is passed to a daemon record and the record is processed inside the JavaIOC, and a RDB query is performed to get an item name list with properties, which satisfied the search constrains. The value is returned back to the client through a dynamically created pvRecord.

One use case of this service is to get a list of EPICS channel names. Since a channel name is an entry of properties for an element, by getting the list back to client, user can retrieve the element's channel names easily.

Gather Service

The Gather Service is another service under pvService module. Basic idea of this service is that a client sends a PV list with a string to this service; the service then creates a pvRecord dynamically with the string name given by the client.

Here we have to mention that the type of each PV in the PV list should have same data type, and pvService does not check it. Also the client has to make sure that name string is unique and did not exist in the Gather Service. Otherwise, it will use existing pvRecord instead of creating a new one. This has to be improved later.

After a client ships a PV list to the gather service, the gather service creates a pvRecord as mentioned above, and connects to low level hardware IOCs for example BPM IOCs, and update its value every time a PV in a low level IOC changes.

Client can customize the Gather as desired service such as a BPM orbit server, or a magnet server.

COMMUNICATION PROTOCOL

An adequate communication protocol is indispensable for SOA architecture. There are many protocols available such as HTTP, XML-RPC and so on. A new generation of EPICS Channel Access protocol, pvAccess, is a better option to deliver accelerator data over the network. The main advantages are as below:

- It fully supports pvData, and depends only on project pvData. We can integrate our servers seamlessly with pvData.
- It is developed against current Channel Access, and inherits the advantages of EPICS Channel Access. For example, it is data stream oriented protocol, and can be expected to have good performance for an accelerator control system.

The performance benchmarking is undergoing, and a preliminary result shows a good performance. For example, on a local office network, when we feed 1000 PVs to the Gather Service, it can update the 1000 PVs' value with a frequency large than 100Hz.

PLAN

Some service such as Simulation Service, itemFinder, and gather service are being prototyped. They all are in the stage of choosing a good communication protocol for production and EPICS pvAccess shows a good performance as communication protocol. Some more development and benchmarking are necessary for a production server.

ACKNOWLEDGEMENT

The authors would like to thank Matej Sekoranja at COSYLAB and Marty Kraimer for their contributions on epics-pvdata development. They also want express their thanks to Ji Qiang at LBNL for providing IMPACT-T code. They want to give their thanks to Leo Dalesio at BNL for his continuous support and encouragement.

REFERENCES

[1] J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey, "A Three-Dimensional Quasi-Static Model for High Brightness Beam Dynamics simulation", Phys. Rev. ST Accel. Beams **9**, 044204 (2006).

[2] P. Chu *et al*, "Generic Model Host System Design", Proc. of IPAC10, TUPEC071

[3] G. Shen *et al*, "Design of Accelerator Online Simulator Server Using Structured Data", Proc. of IPAC10, WEPEB024

[4] http://sourceforge.net/projects/epics-pvdata/

PRESENTATION ONLY

PRESENTATION ONLY

PRESENTATION ONLY

synApps: EPICS APPLICATION SOFTWARE FOR SYNCHROTRON BEAMLINES AND LABORATORIES*

T. M. Mooney#, ANL, Argonne, IL 60439, U.S.A.

Abstract

synApps[1] is a collection of EPICS [2] application software originally intended to support the needs of scientists performing experiments at synchrotron-radiation beamlines. The collection contains general-purpose software that extends or exploits capabilities of EPICS base, and a large amount of instrument-specific software that uses EPICS to control and provide a user interface for off-the-shelf electronics.

This paper will provide an overview of synApps, describe how the software is deployed at the Advanced Photon Source, and highlight recent additions.

OVERVIEW

synApps is a collection of EPICS modules that supplement the record types, device support, and other software infrastructure included in EPICS Base. Because it was written to support scientists conducting a wide variety of experiments, most of the software in synApps is general in purpose, and was engineered to serve many needs at once, by abstracting from specific sets of requirements general solutions for classes of problems.

But this focus on general solutions does not distinguish synApps from other EPICS-application software. Most EPICS software is general purpose, in part because EPICS is a collaborative effort. synApps differs from mainstream EPICS-application software in three ways: it contains a small amount of synchrotron-specific software, it provides infrastructure to support run-time programming, and it provides infrastructure to support data acquisition.

synApps consists of the following modules, grouped according to the kinds of applications they support.

General-Purpose Modules

- **autosave** – Saves the values of EPICS process variables, and restores them after a reboot.
- **busy** – Extends EPICS' execution tracing to include client software.
- **calc** – Provides variations of the EPICS *calcout* record for systems of expressions (*transform* record), string expressions (*sCalcout* record), and arrays (*aCalcout* record).
- **sscan** – Supports *scans* (systematically set conditions; acquire and store data).
- **std** – Supports scalers, sequences of operations, and PID loops.

Hardware Specific Modules

- **areaDetector** – Supports multidimensional detectors.
- **camac** – Supports CAMAC hardware.
- **dac128V** – Supports an IndustryPack digital-to-analog converter.
- **delayGen** – Supports delay generators.
- **dxp** – Supports DXP digital-signal processing spectroscopy systems.
- **ebrick** – Supports the EPICS Brick, a PC104-based computer running Linux, as an EPICS *IOC* (Input/Output Controller).
- **ip** – Supports various message-based (e.g., serial, GPIB) devices.
- **ip330** – Supports an IndustryPack analog-to-digital converter.
- **ipUnidig** – Supports an IndustryPack digital I/O module.
- **love** – Supports Love controllers.
- **mca** – Supports multichannel analyzers and multichannel scalers.
- **modbus** – Supports Modbus devices.
- **motor** – Supports stepper and servo motors.
- **quadEM** – Supports a four-channel electrometer.
- **softGlue** – Provides user-programmed digital logic and I/O.
- **vac** – Supports vacuum-related devices.
- **vme** – Supports VME hardware.

Synchrotron-Radiation Specific Modules

- **optics** – Supports X-ray monochromators, slits, optical tables, and other synchrotron-radiation equipment.

Other Software in synApps

- **xxx** – Provides a template for an EPICS IOC directory using synApps.
- **utils** – Provides miscellaneous software related to synApps, including support for migrating from one version of synApps to another, support for a data-file format used by synApps scan software, and support for rapid EPICS-database programming.

Software Distributed with synApps

synApps makes use of the following EPICS modules that are not part of synApps, but are distributed with it: **allenBradley**, **asyn**, **ipac, seq**, **stream**, and **vxStats**.

#mooney@aps.anl.gov

RUN-TIME PROGRAMMING

Because many synApps users are scientists conducting experiments, and because experimental work is typically less well understood in advance than are other activities supported by EPICS, synApps places a much greater emphasis on support for programming at run time than is typical of EPICS-application software. Most synApps IOCs load records reserved for run-time programming; and most synApps record types are supported by displays, online documentation, and autosave-request files for this purpose. Also, many synApps record types check and report to the user the states of their link fields, so that run-time link errors can be recognized promptly.

In this context, "programming" does not mean code development or scripting, but rather the configuration and linking together of EPICS records. A collection of linked EPICS records – an EPICS *database* – can be viewed as a program in a very high-level language. For example, an input record linked through a calculation record to an output record can implement a feedback loop.

An EPICS database configured at run time is not distinguishable in any essential way from a similar database configured at build time: it has the same speed and efficiency, and it can drive or be driven in the same ways. Thus, run-time-programmed databases can be layered, sequenced, event driven, or scanned, and the result for end users is an extraordinarily powerful and versatile capability to diagnose and solve problems as they arise during an experiment, and to modify solutions to those problems as they become better understood.

The principal means by which run-time programming is accomplished in EPICS is the redefinition of an EPICS link. In early versions of EPICS, links could not be changed at run time. The first programmable links were implemented by Marty Kraimer for use by the synApps *scan* and *wait* records (originally developed by Ned Arnold), and they were initially viewed as support for scans. But the *wait* record quickly came to be applied more widely for its run-time programming capability, and the result was powerful enough to motivate the development (by Marty Kraimer, Bob Dalesio, Jeff Hill, and others) of support for run-time redefinition of *all* EPICS links.

The impact of run-time-programmable links on synApps' development was profound: most synApps record types, databases, and displays came to be developed with run-time programming as an objective, and the automated saving and restoring of EPICS PV values (*autosave*, originally developed by Bob Dalesio) acquired new urgency and purpose.

Recently, the notion of run-time programming was extended to run-time development of digital hardware, in the **softGlue** module.

Rapid Prototyping

Soon after support for run-time programming became pervasive in synApps, the capability was recognized also as a rapid-prototyping tool – a way for EPICS-database developers to test and combine database fragments without rebooting. The principle defect in this development approach was the lack of a convenient way to save run-time programming in the standard form of an EPICS database file.

A wxPython program, *snapDb*, was written to address this problem. Using snapDb, a user or developer can produce a loadable EPICS database from run-time-programmed fragments simply by using MEDM's Drag-And-Drop capability to enter a PV name from each record into a list. snapDb then reads all fields of the listed records, and writes an EPICS database file. snapDb can also write an MEDM display file for the database.

DATA ACQUISITION

Synchrotron-radiation users spend a lot of time scanning – systematically varying conditions, acquiring data under those conditions, and storing the data for later analysis. The **sscan** module is dedicated almost entirely to this purpose, comprising the *sscan* record, which performs multidimensional scans; the *recDynLink* library, which manages Channel-Access connections for the *sscan* record; and the *saveData* task, which writes scan data to disk.

Other synApps modules involved heavily in data acquisition are the **areaDetector**, **mca**, **dxp**, and **std** modules. These modules support specific hardware, such as scalers, multichannel analyzers, and two-dimensional detectors, and do so in a way that permits EPICS clients, including the *sscan* record, to trigger data acquisition, wait for acquisition to complete, and collect the resulting data.

Completion Reporting

EPICS Base contains support for tracing the execution of a linked set of records (i.e., a *database*), and for signaling the completion of that execution to the client that caused it to occur. Within an IOC, tracing is performed by the EPICS *putNotify* facility. Execution spanning more than one IOC can be traced by using the Channel Access function *ca_put_callback()* to make the completion of a record in one IOC contingent on the completion of execution in another IOC.

synApps' data-acquisition strategy relies heavily on *putNotify* execution tracing, and synApps provides several record types engineered to extend *putNotify* across IOCs, in addition to serving their primary purposes:

- ***sscan*** – This record performs a one-dimensional scan. Several ***sscan*** records can be linked to perform multidimensional scans.
- ***sseq*** – This record is a variant of the EPICS ***seq*** record, which performs a programmed sequence of operations. The ***sseq*** record differs from ***seq*** in that it can read and write strings as well as numbers, and it can wait for completion between operations.
- ***swait*** – This record is an early prototype of the EPICS ***calcout*** record, and is one of the first EPICS records whose links could be modified at run time. It differs from ***calcout*** in that it uses the *recDynLink* library, and its output link waits for completion.

- ***aCalcout*** – The array calcout record is a variant of the EPICS ***calcout*** record, and differs from it by supporting array fields and expressions in addition to scalar fields and expressions. The ***aCalcout*** record also can wait for completion of execution triggered by its output link.
- ***sCalcout*** – This record is similar to the *aCalcout* record, but it supports strings, instead of arrays.
- ***busy*** – This record functions as a proxy for the execution performed by a Channel Access client. EPICS *putNotify* cannot directly trace execution by a client, so the busy record (which can be traced) pretends to be executing until the client tells it to stop.

Most of the listed record types have links that can use the Channel Access function, ca_put_callback(), to initiate execution, and that can wait for the resulting callback, which indicates that the execution has completed. The *busy* record is an exception: its purpose is to be *driven* by a ca_put_callback(), and to look busy until a client tells it to stop, whereupon its completion yields a callback indicating that the client is done.

Automated testing

The infrastructure with which synApps supports data acquisition by users is also useful to developers, for diagnostic and testing purposes. The *sscan* record, for example, has been used (with other run-time configured software) to diagnose race conditions, by systematically varying the time between the execution of application code, and a simulated response from driven equipment.

The combination of the *sscan* record and the **softGlue** module extends this diagnostic and testing capability to digital hardware.

DEPLOYMENT AT APS

The deployment of synApps at the Advanced Photon Source has evolved in response to an increasing number of beamlines, an increasing emphasis on computer security, and the similarly driven evolution of the EPICS module structure. Originally, synApps modules (called "Apps" in those days) were deployed alongside IOC directories on a file server to which beamlines had read/write access; there was not a clear distinction between support modules and application modules.

As the number of beamlines increased, and the separation between beamline subnets became more complete and more rigidly enforced, synApps was split into support modules and IOC directories. Support modules (all modules except **xxx**) are now hosted, along with EPICS Base, on a central file server, and both are distributed via *rsync* to read-only partitions on secondary servers dedicated to individual beamlines. The IOC directories are now created on read-write partitions of those secondary servers, and begin as copies of the synApps **xxx** module, which collects support from all other synApps modules and builds loadable executables and database-definition files for use by one or more IOCs.

One effect of this evolution has been the concentration of display files and autosave-request files in support modules, rather than in application directories. In turn, this concentration led to the development of an include-file capability in autosave, so that module developers could define the PVs needed to restore databases implemented in those modules, and IOC directories could simply include the request files for the databases they needed to maintain through IOC reboots.

Another effect has been an increasing reliance on MEDM's ability to build displays using *Composite Objects* – display files that can be included within other display files and customized using macro substitution.

RECENT DEVELOPMENTS

areaDetector

The **areaDetector** module provides a general-purpose interface for area (2-D) detectors in EPICS. It supports a wide variety of detectors and cameras, ranging from high-frame-rate CCD and CMOS cameras, pixel-array detectors such as the Pilatus, and large-format detectors like the MAR-345 online imaging plate.

Among recent improvements in **areaDetector** is the evolution of support for plug-ins, which provide a mechanism for device-independent real-time data analysis, such as regions-of-interest and statistics.

softGlue

The **softGlue** module provides EPICS users and developers with the capability of creating and modifying simple digital electronic circuits, connecting those circuits to external devices, and controlling or driving the circuits – all by writing to EPICS process variables.

ACKNOWLEDGMENTS

synApps is the product of a collaboration including more people than can practically be named here. Most of the EPICS developers at the Advanced Photon Source – in particular, those in the Beamline Controls and Data Acquisition Group – and many at other EPICS sites, have contributed code or ideas, tested on or ported to new architectures, improved the build software, etc. Mark Rivers is responsible for much of the recent work in synApps, and maintains a large fraction of synApps, including the **areaDetector**, **camac**, **dac128V**, **dxp**, **ip330**, **ipUnidig**, **mca**, **modbus**, and **quadEM** modules.

REFERENCES

[1] http://www.aps.anl.gov/bcda/synApps.
[2] http://www.aps.anl.gov/epics.

USING EZCAIDL TO CONNECT TO EPICS CHANNEL ACCESS FROM SHADOWVUI FOR DYNAMIC X-RAY TRACING*

Alan Duffy#, Canadian Light Source Inc., Saskatoon, Saskatchewan, Canada

Abstract

Using the ezcaIDL library, for IDL [1], to provide an interface to EPICS [2] Channel Access through the EZCA [3] library, a simple XOP [4] extension was written that initializes ezcaIDL and thus allows access to a set of simplified IDL interface commands to connect to Channel Access from within XOP and hence from SHADOWVUI (an XOP extension) [5]. The XOP widget-based driver program is a commonly used front-end interface for computer codes of interest to the synchrotron radiation community. It models x-ray sources and characterizes optics. Extensions, such as SHADOWVUI, are optionally loaded to easily expand its functionality. SHADOWVUI is a complete Visual User Interface for SHADOW [6], which is an essential tool for x-ray optics calculations and ray-tracing. SHADOWVUI is an interactive tool for designing an optical system and visualizing results as graphs and histograms. The working scheme is to define the source and the optical elements by entering their parameters. The author has taken the usual SHADOWVUI simulation of an x-ray system a step further by using ezcaIDL to interface with the EPICS control system to access the positions of optical components in real life and then run a corresponding simulation based upon these.

INTRODUCTION

In order to predict the performance of an optical system in general and in particular a synchrotron radiation beamline, ray tracing methods are used. An essential tool for x-ray optics calculations is the ray-tracing program SHADOW, developed at Nanotech Wisconsin (University of Wisconsin), and has been used in the synchrotron community during the last 20 years. A complete Visual User Interface for SHADOW aptly named SHADOWVUI may be used as a higher level interface with graphics and menus to prepare the SHADOW inputs. It is available as an extension to another commonly used software package called XOP, a commonly used front-end interface for computer codes that model x-ray sources and optics. Essentially, the SHADOW inputs define the optical system as a collection of optical elements (mirrors, slits, screens, etc.) placed in sequential order. SHADOW generates and traces a beam from the source (e.g. bending magnet, wiggler, or indulator) sequentially through the system. The important point is that the SHADOW inputs define the optical system which usually serves to model a real synchrotron beamline. However, the parameters are static and do not change until the user enters new ones.

CONCEPT

The concept of running a dynamic x-ray tracing simulation of a beamline is straightforward (take the live positions and put them in the simulation engine), but requires some preliminary work creating the model in SHADOWVUI and determining the corresponding inputs to use from the actual beamline. This involves defining the source by supplying its parameters (e.g. energy, etc.) and defining the various optical elements with their parameters (e.g. mirror types, source plane distances, image plane distances, etc.), and how they relate to beamline parameters. The ezcaIDL library provides the tool necessary to read the beamline parameters that are maintained by the EPICS control system. The only catch is that one must define how the variables in the model are related to the parameters of the beamline. The newly developed XOP extension is used in conjunction with SHADOWVUI and requires as input a user created IDL structure defining the relationship between beamline parameters (*i.e.* process variables) and SHADOWVUI variables to make connections between the live position of the beamline optics and the variables in the simulation model.

Positioning Optical Elements

The position of each optical element in SHADOW is defined relative to the previous element (or source), not the laboratory reference frame. The user inputs the incidence and reflection angles of the central ray at each optical element as well as source and image distances to define the system. In an aligned system the central ray coincides with the optical axis, however the user has complete freedom of specifying incidence angles that are zero, positive, or larger than 90 degrees, as long as the user understands how to interpret the results. It is also not necessary for the image and source distances to correlate to the location of an actual image or object in the optical sense either. The sum of the image distance (from the previous element) and source distance simply defines the separation between optical elements in the SHADOW model. In fact, it is advantageous to think of these distances not as defining the optical element positions *per se*, but as defining the origins of their coordinate systems. Then use the mirror movement option available to place the optical components in their proper locations. Using this option to place an optical element prevents unnecessarily moving subsequent components with their positions defined relative to previous components and avoids having to recalculate distances and angles.

*Work supported by CLS
#alan.duffy@lightsource.ca

Beamline Parameters and Model Variables

The relationship between beamline parameters and the SHADOW model variables is of key importance for the dynamic simulation. A parameter in the model may be determined by one or more process variables, and vice versa. The simplest situation would be a one to one relationship between a model variable and a beamline parameter (process variable). In fact, by creating an appropriate EPICS record, one could have a single PV for each model variable used in the simulation. In any case, the user provides the relationship between the model variables and process variables as strings defining the equation(s) that relates them as part of a PV_INFO structure as illustrated in Fig. 1.

```
«struct»
ENERGYFEEDBACK : PV_INFO
pv = 'BL1606-B1-1:Energy:fbk'
desc = 'Beamline Energy [eV]'
pv_min = 1500
pv_max = 10000
oe_num = 0
src_num = 0
pv_2vui =
'(*ptrSRC).PH1 = beamline.EnergyFeedback.vui_val
& (*ptrSRC).PH2 = beamline.EnergyFeedback.vui_val'
vui_2pv = '0.5*((*ptrSRC).PH1+(*ptrSRC).PH2)'
vui_val = 0.0
```

Figure 1: Example PV_INFO structure.

All the information needed for the simulation is stored in an IDL structure named beamline with a field for each process variable containing a nested PV_INFO structure. The fields of this nested structure are described in table 1.

Table 1: PV_INFO structure content

Field	Type	Description
pv	string	EPICS process variable string
desc	string	Text to describe process variable
pv_min	float	Lower limit
pv_max	float	Upper limit
oe_num	int	Optical element number (zero otherwise)
src_num	int	Screen number (zero otherwise)
pv_2vui	string	Equation(s) to convert value of PV(s) to SHADOWVUI variable
vui_2pv	string	To convert value of SHADOWVUI variables(s) to PV value
vui_val	float	Stores SHADOWVUI variable value

In the example show in Fig. 1, the beamline parameter is the energy feedback. The process variable string 'BL1606-B1-1:Energy:fbk,' is stored and retrieved from the IDL variable *beamline.ENERGYFEEDBACK.pv*, and so on. A SHADOWVUI workspace stores its parameters in a structure in a state variable that includes variables one finds in the start.xx files. The start.00 file, for example may have the line: *PH1 = 5000.00000.* The corresponding SHADOWVUI variable is the rather obtuse looking (*(state.ptrsrc)[state.ifc.src_sel]).PH1. In order to simplify things, the ezcaShadowVUI extension defines ptrSRC, ptrOE1, ptrOE2… as pointers to the source parameters, and optical element parameters. As these are pointers, the dereference operator '*' must be used as appropriate. For example the bend radius of optical element 1 is *(*ptrOE1).RMIRR*.

The entire structure with all the information for the process variables that are to be incorporated into the simulation must be defined. This may be done by creating the structure in an IDL session and then saving it to an IDL file such as *pv_defs.sav*, or by creating a file with the commands to define the structure and executing it from SHADOWVUI using the *xop_macro_compact* command. The former method requires the user to restore the .sav file, with the command: *restore, 'pv_defs.sav'*. Once the variable is created or restored in a SHADOWVUI macro, the user can then call: *reshadowvui, beamline.* This will start up a widget similar to the one in Fig. 2.

EZCASHADOWVUI WIDGET

The ezcaShadowVUI widget has a tab for the source, and each optical element, and sub-tabs for each screen. The widget shows live process variable values and the equivalent SHADOWVUI value calculated from the vui_2pv string field.

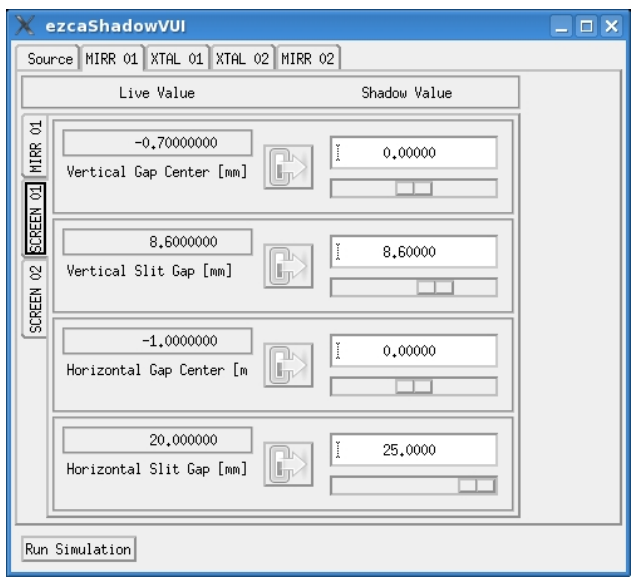


Figure 2: ezcaShadowVUI widget interface.

The ezcaShadowVUI extension uses the ezcaIDL channel access features to set up process variable

monitors so that the live values update once per second. The equivalent SHADOWVUI value is shown in an editable box with a slider. The user can click a button to copy the live value to SHADOWVUI, doing so executes the command stored in the pv_2vui string field. After the user has finished copying live values and/or editing the SHADOWVUI values, the *Run Simulation* button may be pressed to execute the source generation and ray trace routines and to show a plot of the beam focus. Also the ezcaShadowVUI widget is non-modal, meaning that the user can do other things with the main SHADOWVUI interface while this widget is running. In particular, the user still has access to all the usual features of SHADOWVUI, such as running macros. The widget code can be modified to allow automatically running a macro to be triggered either by the *Run Simulation* button or a process variable event.

Dynamic Ray Tracing

The copying of live values to the simulation engine, as well as running the source generation and a ray trace with a new plot, may be set up to be down automatically for true dynamic ray tracing. There is a limitation, however, on the rate at which the ray tracing can occur. It is possible to decrease the number of rays to increase the simulation speed, but this may not be desirable. However, a refresh rate of a new trace every few seconds should not pose a problem.

Installation

The installation of the XOP extension ezcaShadowVUI is done by creating an ezcashadowvui folder in the XOP extensions directory and copying the ezcashadowvui.sav file into it. EPICS should be installed with the extensions [3] ezca, ezcaIDL, and EzcaScan so that the appropriate libraries are available. Finally, the EZCA_IDL_SHARE environment variable should point to the location of the libezcaIDL.so file. The ezcaShadowVUI extension will be made available in the near future.

CONCLUSION

The ezcaShadowVUI extension is in its preliminary stage of development, but has proven useful in modelling a real beamline, the SXRMB beamline at the Canadian Light Source, and should prove useful in modelling other beamlines. The most time consuming part of setting up the dynamic simulation is determining the relationship between SHADOW variables and beamline parameters and then creating the structure to contain that information. However, it is just a matter of reconciling the different coordinates systems of one with the other. The real challenge as always is to properly understand the beamline and in particular the intricate details of pivot points and rotation axis. In order to have a truly accurate simulation, it is important to be attentive to these details.

REFERENCES

[1] Interactive Data Language (IDL) by ITT Visual Information Solutions, 4990 Pearl East Circle, Boulder Colorado 80301, United States of America; http://www.ittvis.com/.

[2] http://www.aps.anl.gov/epics/.

[3] http://www.aps.anl.gov/epics/download/extensions/

[4] Roger J. Dejus and Manuel Sanchez del Rio, "XOP: a multiplatform graphical user interface for synchrotron radiation spectral and optics calculations," Proc. SPIE, Vol. 3152, 148 (1997).

[5] http://www.esrf.eu/UsersAndScience/Experiments/TBS/SciSoft/xop2.3/extensions/

[6] C. Welnak, G.J. Chena, and F. Cerrinaa, "SHADOW: A synchrotron radiation and X-ray optics simulation tool," Nucl. Instr. and Meth. in Phys. Res. A, Volume 347, Issues 1-3, 11 August 1994, Pages 344-347

A SIMPLE DAQ SYSTEM BASED ON LABVIEW, PHP AND MYSQL

M. Tanigaki*, K. Takamiya, R. Okumura,
Research Reactor Institute, Kyoto University, Kumatori, Osaka 590-0494 Japan

Abstract

A tiny and simple DAQ system has been designed and developed for the application to the control system in our institute. This DAQ system is based on on LabView, MySQL and apache, and shows good compatibility with LabVIEW-based system like the control system for the FFAG complex in our institute. The current status for the development, as well as the recent accelerator-related status in our institute, will be introduced.

INTRODUCTION

An FFAG accelerator complex[1, 2, 3] has been developed as a proton driver for the feasibility study on ADS performed in the research reactor institute, Kyoto University. The control system for this FFAG accelerator complex has some requirements on the flexibility, simplicity and reliability. The control system is required to have a sufficient flexibility towards major and minor modifications in the design and equipments of accelerator complex during the construction, and to achieve a certain level of easiness on its use and development for the people in our institute, who are little familiar to accelerator itself. Additionally, high reliability and stability from the points of the nuclear safety and the radiation protection are required since the combined operation with a nuclear fuel assembly is planned in the feasibility study on ADS.

To meet such requirements for the present control system, we have developed a control system [4] based on LabVIEW, known as its user-friendly GUI environment, and PLC known as one of the most reliable control devices in the field of factory automation. This control system for the FFAG complex has proven itself to have sufficient performance and to satisfy the requirements on the design through the construction and operation of the FFAG accelerator complex, in its operation for years. Based on this success, this control system has been applied to other equipments and facilities. One of such typical examples is that the application to the pneumatic transportation facility in KUR[5].

On contrary to the control system itself, little efforts have been made for the data acquisition system up to now. In most of the application cases, a simple data logging feature is included in VIs by using the functions of LabVIEW such as the chart VI. As the increasing demand on the systematic management of the data for the multiple devices and on the simplified method of DAQ for the users, we have started the development of a DAQ system for our control system.

* tanigaki@rri.kyoto-u.ac.jp

In this paper, the outline and current status of our DAQ scheme are introduced.

DAQ SYSTEM WITH ODBC DRIVER

At present, the FFAG accelerator complex in our institute is under modification to the injection scheme using H^- beam. The FFAG injector will be replaced to an 11 MeV H^- proton linac by the end of the fiscal year 2010. Additionally, the inclusion of the present control system to a new control system based on EPICS, intending to the inclusion of this FFAG accelerator to a larger accelerator complex for the pulsed neutron source. Therefore, the main application of the control & DAQ system for now is the devices and instruments equipped to the 5 MW reactor, especially the pneumatic transportation facility for the neutron irradiation [5].

The outline of the pneumatic transportation apparatus and the control system is shown in Fig. 1. The control system for this pneumatic transport system is the same architecture as that for the FFAG complex [4]. The low level sequences of PLCs for controlling the pneumatic transportation system has been implemented in PLCs, and the man-machine interfaces (MMIs) are developed with LabVIEW on conventional PCs. In addition to the controlling system of the pneumatic transportation apparatus and the monitoring system, related external systems such as radiation control systems and measurement systems for experiments are integrated. This integrated system might well be able to realize secure operating and management of the pneumatic transportation apparatus.

In this pneumatic transport system, a DAQ system based on the ODBC driver, LabVIEW and MySQL is developed. So called, a "SQL Command Generator" VI is implemented into every MMI PC as a sub VI of MMI VIs. Since

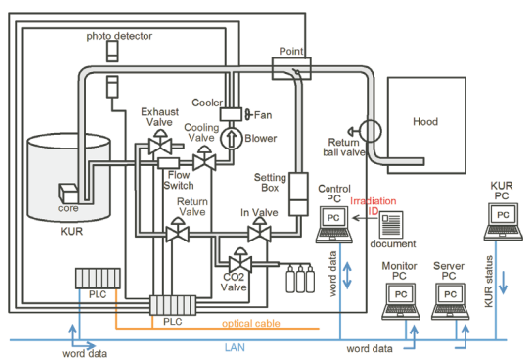


Figure 1: Outline of the pneumatic transportation apparatus and the new control system.

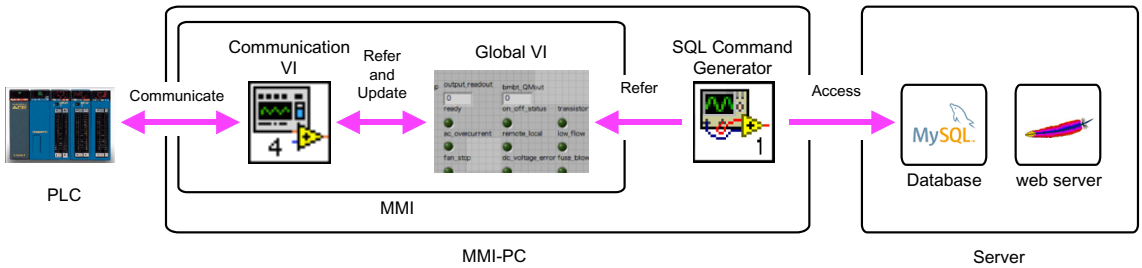
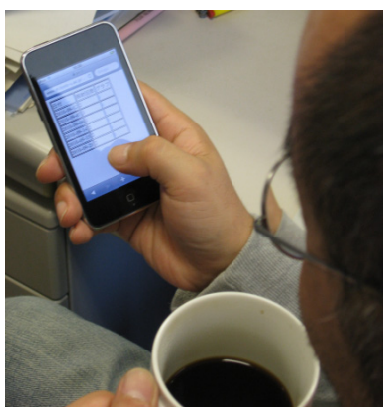


Figure 2: Architecture of the DAQ for the pneumatic transport system at KURRI. The DAQ software is implemented as a sub-VI to the existing MMI software for the control system. All the parameters are stored in the MySQL database and these data are available to users through the web server.

(a) Start page of the data browsing.

(b) A typical example of data browsing stored in the SQL server. In the present data browsing system, the graphs are generated by "PHP/SWF Charts" available from maani.us[6].



(c) Registered users are allowed to browse the status via conventional web browsers.

Figure 3: Screen shots of the web-based data browsing system.

all parameters are expanded on the global VIs in MMI PCs in this control system, each implemented SQL Command Generator VI obtains the status and information of connected devices by referring via respective Global VIs, and send the SQL command by use of a MySQL ODBC driver to pass the data to the remote MySQL database server. The database server stores and manages various data, such as the start/stop time of irradiation, the pressure of CO_2 gas in the transport tube and the radiation levels in the pneumatic transport facilities, on a database constructed on a MySQL server.

For the access by users to the data stored in the respective database, Apache, which is the most popular web server application, is also installed in the server. The stored data can be accessed through the internet using common web-browsers installed in PCs or recent mobile phones (Fig. 3). Accessing logged data over the internet and receiving warning messages by e-mail are enabled in the developed systems.

DAQ SYSTEM WITH POST METHOD AND PHP

The present DAQ system for the pneumatic transport system works quite fine, but the application to other facilities are not straightforward because of the direct implementation of ODBC drivers into MMI software. This procedure requires the developers to handle SQL commands for the DAQ system. For example, once a developer decides to add another parameter to be recorded, they have to treat additional SQL commands.

In the common web-based services, the parameters in these services are often send along with their names by POST method, and processed by php scripts and stored in databases on SQL servers. In the control systems in KURRI, allocation tables for the parameters are defined and each parameter can be uniquely assigned by the names of the equipment and its respective parameter. Therefore, we are able to apply such conventional POST method procedure to our DAQ system.

The architecture of the php-based DAQ system being currently developed is shown in Fig. 4. This DAQ VI obtains all of the parameters from PLCs in the same commu-

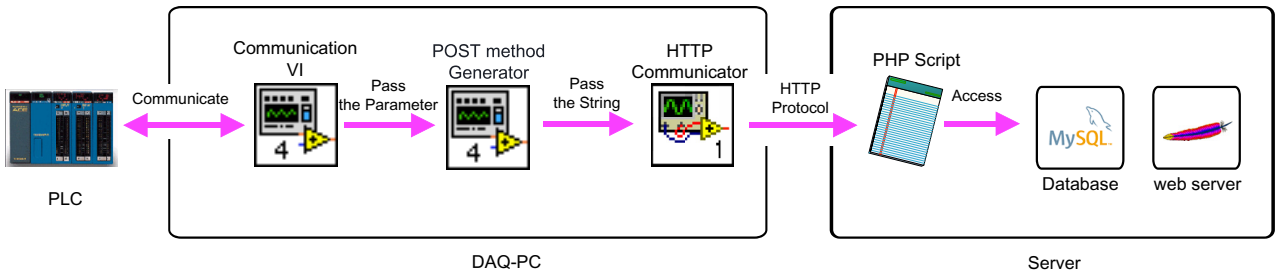


Figure 4: Architecture of the php based DAQ system developed at KURRI. Unlike the present DAQ for the pneumatic system, this DAQ system works as a stand-alone software. The web server is used not only for the data browsing by users, but also for processing the data transfer based on the HTTP protocol.

nication procedure as the communication VI uses. Then the parameters are converted to a set of chunks, in which the device name, parameter name and its value are sequentially listed. This DAQ VI also accesses a web page on an remote web server, in which the php script for the data processing is implemented. The data is then transferred by the POST method, and processed by the php script based on their assigned names to be stored in a table on a MySQL database. One table is usually prepared on a MySQL database for each device, and the proper tables are selected by the name of devices given in the is usually prepared by the equipment base. As long as the developers follows the same allocation table as used in the control system, the parameters can be stored without any initial settings except creating a corresponding table on a database with this scheme.

CURRENT STATUS

As for the DAQ systems discussed above, the DAQ system for the pneumatic transport system has been served for the actual experiments of neutron irradiation as soon as our 5 MW reactor resumed the operation in June 2010, and no troubles arising from the DAQ system are reported up to now. The php-based system has been almost finished the evaluation period, and the actual implementation is expected soon.

As for our institute, another accelerator project may be expected in near future. Recently, our research project "Promotion of Leading Research toward Effective Utilization of Multidisciplinary Nuclear Science and Technology", which has the construction of an accelerator-driven neutron source as the key facility, is included in the list of recommended large projects to the Japanese government. This means we may have a possibility to build another proton/heavy ion accelerator with its energy of $30 \sim 100$ MeV in our institute. We have started the re-organization of the developed VIs and ladder sequences for our FFAG accelerator complex and other equipment in our institute, as well as the training of the technicians in our institute for the expected developments on the control system.

References

[1] D. Normile, Science **302**, 379 (2003).

[2] M. Tanigaki, K. Mishima, S. Shiroya, Y. Ishi, S. Fukumoto, S. Machida, Y. Mori, and M. Inoue, in *Proceedings of EPAC 2004* (Lucerne, Switzerland, 2004), pp. 2676–2678.

[3] T. Uesugi, Y. Mori, H. Horii, Y. Kuriyama, K. Mishima, A. Osanai, T. Planche, S. Shiroya, M. Tanigaki, K. Okabe, et al., in *Proceedings of EPAC08* (Genoa, Italy, 2008), pp. 1013–1015.

[4] M. Tanigaki, K. Takamiya, H. Yoshino, N. Abe, T. Takeshita, and A. Osanai, Nuclear Instruments and Methods A **612**, 354 (2010).

[5] K. Takamiya, R. Okumura, N. Abe, Y. Nakano, K. Miyata, S. Fukutani, A. Taniguchi, and H. Yamana, JOURNAL OF RADIOANALYTICAL AND NUCLEAR CHEMISTRY **278**, 719 (2008).

[6] maani.us, `http://www.maani.us`.

WEB SERVICES CYBER-SECURITY ISSUES*

D. Quock#, ANL, Argonne, IL 60439, U.S.A.

Abstract

The Web's potential for distributed programming has been proven not only in the business realm, but also in the accelerator controls domain. Web services describes clients and servers that communicate over the Internet's Hypertext Transfer Protocol (HTTP) using predefined Internet-based Application Programming Interfaces (APIs). It is the uniqueness of Web services transactions such as cloud computing, data sharing, and data archiving that give rise to the security concerns of Web services (authentication, data integrity, non-repudiation, and privacy). At Argonne National Laboratory's Advanced Photon Source, Simple Object Access Protocol (SOAP)-based Web services were implemented into the Integrated Relational Model of Installed Systems (IRMIS) as the application interface to Oracle's Content Server document management software. This report reviews the basics of Web services, cyber-security issues that are inherent for Web services, current Web services security implementation practices, and future directions of Web service security development efforts where the overriding goal of Web services security is to focus on managing risk and protecting data.

BASICS OF WEB SERVICES

In simplest terms, Web services are distributed Internet applications that have standard-based interfaces. Web services are typically thought of as being divided into two main technologies:

1. Big Web Services: This technology uses Extensible Markup Language (XML) messages that follow the Simple Object Access Protocol (SOAP) standard.

2. RESTful Web Services: The representational state transfer (REST) software architecture uses PUT, GET, DELETE and POST HTTP methods to integrate Web browsers with underlying client/server software applications.

Service-Oriented Architecture (SOA) is model-based software that is typically constructed from loosely coupled Web services. SOA can be broken down into the three layers: business workflow, Web services, and communication [1]. Table 1 demonstrates that SOA adds three layers on top of the standard client-server architecture and shows the associated Web services standards that are used at each layer.

Table 1: SOA Architectural Layers

Architectural Layer	Web Services Standards
Business Workflow	BPEL
	WSCI
Web Services	WSDL
	UDDI
Communications	
SOAP	XML
Client-Server Transports	
HTTP	
SSL	
TCP/IP	

Web Services Standards

At the highest level of SOA, Business Process Execution Language (BPEL) is used to describe and execute the business processes. An alternative to BPEL is the World Wide Web Consortium (W3C)'s standard Web Service Choreography Interface (WSCI). These two standards are currently diverging as industry is divided in its support of either business workflow standard. The role that BPEL (or WSCI) plays in SOA is orchestrating the overall business workflow by providing mapping between the services and business processes through documents.

At the next level of SOA, the standard Web Service Description Language (WSDL) provides static interface definitions for the software components that are accessible to clients. The Universal Description, Discovery and Integration (UDDI) is a specification for repositories where organizations can publish services that they provide and describe the interfaces to their services via WSDLs.

At the communications layer of SOA, messages are transmitted through SOAP, which is an envelope containing a header and body. The services that are communicating with each other can be identified through their unique name contained in SOAP messages.

ADVANCED PHOTON SOURCE WEB SERVICES

The benefits of SOA to organizations is the flexibility of implementing business processes on top of Web services and the ability to compose and re-compose systems frequently. SOA provides a peer-to-peer style of architecture with a general statelessness of services. One example of how Web services technology was implemented at Argonne National Laboratory's Advanced Photon Source is in the interaction between the in-house built IRMIS accelerator controls relational database

*Work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

#quock@aps.anl.gov

software application and Oracle Content Server document management software.

Figure 1 shows an IRMIS Applications Organizing Index display that has retrieved general document information for documents stored in Oracle Content Server (ICMS). The search for documents in Content Server was done in IRMIS using a simple PHP SOAP client utility that looks similar to:

```
$client = new SoapClient($wsdl,

        array('login'=>$user_name,'password'=>$user_password));
```

where $wsdl is the https Internet address of the location where the SOA WSDL file can be obtained. In this example, the WSDL file defines the interface to a document search software component for Oracle Content Server. The search results are retrieved and stored locally by using the PHP statements:

```
$param = array("queryText"=>$searchString,"sortField"=>"dInDate");
$retVal = $client->AdvancedSearch($param);
$search_results = $retVal->AdvancedSearchResult->SearchResults;
```

Figure 2 shows the home Web page for Oracle Content Server where search options are provided to the user for performing a manual search on its document database. The same search string can be entered into the Content Server's Comment field to obtain the same set of document search results. The benefit of using SOAP Web services provided by Oracle Content Server in the IRMIS PHP application is that the IRMIS user only needs to interface to one software application (IRMIS) to get information provided from two separate database applications. The IRMIS display provides an Internet link to directly launch each document in its native format, thus providing even more ease of use and efficiency to the user. The implementation shown here for Web services in IRMIS is the very simplistic case of client software interacting with a server behind the same firewall in the same organization.

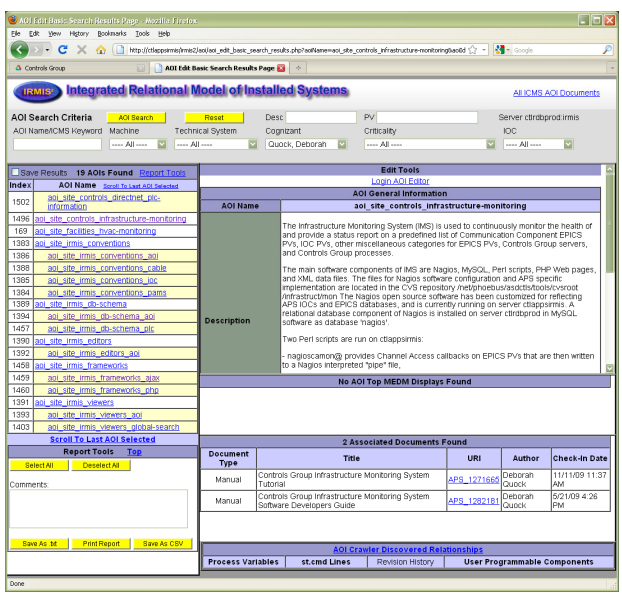


Figure 1: IRMIS display with ICMS search results.

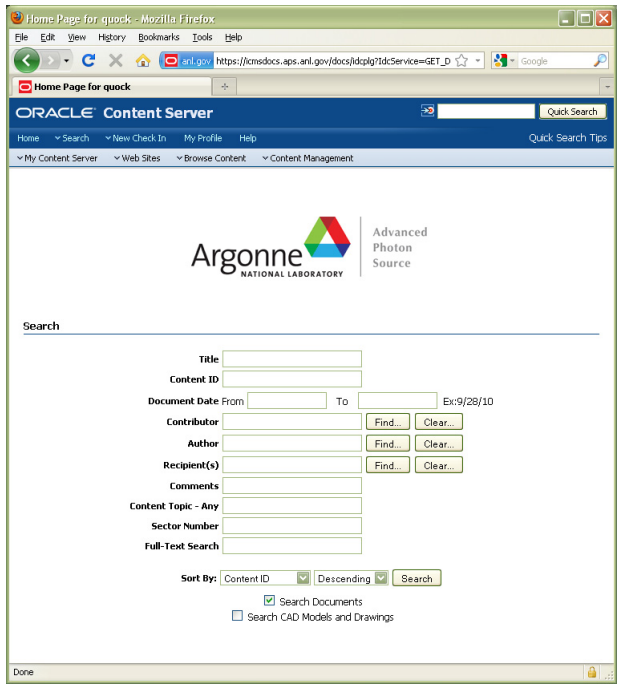


Figure 2: ICMS search home page.

WEB SERVICES CYBER-SECURITY ISSUES

The trade-off for having software components that are defined by their interfaces and can be accessed on the Internet is the increased complexity of the systems where they are used. This increase in complexity in SOA applications is due to several factors including:

- The software used to manage the Web services is complex;
- The boundaries of communication may extend outside of an organization's intranet;
- Dynamic reconfiguration of a client application can be easily obtained through both combination and reuse of individual Web services.

This increase in complexity of a SOA application results in a wider variety of cyber security threats. Examples of such security threats are message alteration, message reading, man-in-the-middle attacks, principal spoofing, forged claims, message replay, and denial of service.

WEB SERVICES CYBER-SECURITY STANDARDS AND ORGANIZATIONS

Web Services Organizations

To effectively deal with cyber security threats that are specific to SOA technology, cooperation and coordination among the vast number of businesses and other institutions using and providing Web service is crucial. The Organization for the Advancement of Structured Information Standards (OASIS) is a not-for-profit consortium that drives the development, convergence and

adoption of open standards for the global information society. The consortium produces more Web services standards than any other organization along with standards for security, e-business, and standardization efforts in the public sector, and for application-specific markets [2]. OASIS sponsors ebXML (Electronic Business using eXtensible Markup Language), a modular suite of specifications. The World Wide Web Consortium is an international community where member organizations, a full-time staff, and the public work together to develop Web standards. Among the many standards developed and supported by W3C are XML Encryption and XML Signature [3]. Corporations such as IBM, Microsoft, and Oracle also contribute to the development of Web services standards.

Web Services Cyber-Security Standards

In addition to the Web services standards list in Table 1, there are several standards specific to addressing cyber-security issues.

- Security Assertion Markup Language (SAML) defines authentication and authorization assertions. SAML assertions can be included in the header or in the payload of a SOAP message.
- WS-* is a general nomenclature used to refer to a family of Web services standards supported by various organizations. Common WS- standards include:
    - WS-Security
      Defines security tokens that can be used for claims of authentication or proof of some right.
    - WS-ReliableMessaging
      Describes a protocol that allows SOAP messages to be reliably delivered between distributed applications in the presence of software component, system, or network failures.

WEB SERVICES DESIGN PRACTICES

For new Web services applications, security considerations should be applied to every aspect of the software development life cycle:

- Security requirements
- Security architecture
- Web services security standards
- Certification (show that software complies with security requirements and security standards)
- Run-time security monitoring
- Penetration testing

The possibilities for implementing software security practices range from the very simple and well-known coding techniques to extremely analytical and involved source code analysis methodologies. A rule of thumb for designing user interfaces is that simple interfaces with few options are easy to test and audit. “Giant APIs require giant security measures” [4]. Another commonly used practice is SSL (Transport Layer Security) for encrypting and verifying the integrity of every client request. Source code analysis tools for identifying security weaknesses include:

- Vulnerability databases that are published to the general public (e.g., Microsoft publishes one);
- Pointer and reflection analysis that constructs a call graph that allows input data to be traced along function calls [1].

There are many Web services security analysis software products available on the market. The functionalities and standards that they typically examine include conformance validation, integrity checks, XML schema validation, XML encryption, XML signature, WS-Security, user authentication, audit, alert, Web services access control, and content inspection. IBM has developed a service-oriented analysis and design process for modeling, analyzing, designing, and producing a SOA application that is based on Java and IBM WebSphere software development tools. Microsoft, Oracle, Sun, and various other companies have also developed Web services design and Web services manager software tools.

CONCLUSION

The complex nature of SOA applications calls for governance of SOA and its underlying Web services technology. Cooperation among industry and Web services standards organizations is crucial to ensure reliable Internet-based business and government processes, and safeguarding of intellectual property and high-security-level government data. Web services standards organizations are well established and have received widespread support and contributions from major computer and IT corporations. Smaller institutions benefit from readily available Web services standards and Web services security products. Theoretical research in Web services security technology is active at many universities and continues to advance the software security design and monitoring tools available to the general public [1]. At Argonne National Laboratory, Web services applications have been deployed successfully as they make efficient use of disparate software applications.

REFERENCES

[1] C. Gutierrez, ‘Web Services Security Development and Architecture: Theoretical and Practical Issues,’ (IGI Global, Hershey, PA: 2010).
[2] OASIS, http://www.oasis-open.org/who/, 2010.
[3] W3C, http://www.w3.org/standards/webofservices/, 2010.
[4] C. Snyder and M. Southwell, ‘Pro PHP Security,’ (Apress, Berkeley, CA: 2005).

REMOTE ACCESS TO THE VESPERS BEAMLINE USING SCIENCE STUDIO*

D. Maxwell#, D. Liu, E. Matias, D. Medrano, CLS, Saskatoon, Canada
M. Bauer, M. Fuller, S. McIntryre, J. Qin, UWO, London, Canada

Abstract

Science Studio is a web portal, and framework, that provides scientists with a platform to collaborate in distributed teams on research projects, and to remotely access the resources of research facilities located across Canada. The primary application for Science Studio is to provide scientists with remote access to the VESPERS beamline at the Canadian Light Source synchrotron in Saskatoon Saskatchewan, and to readily process data from this beamline at the SHARCNET high performance computing facility in London Ontario. The VESPERS beamline is a complex instrument that is composed of many devices, such as valves, motors and detectors, which are all controlled through the low-level EPICS control system. Science Studio implements a simple, intuitive and functional web-based interface to the beamline for device control and data acquisition. The Science Studio experiment management system allows the acquired data to be easily organized and shared with the research team. This paper will provide an overview of the design, implementation and capabilities of the Science Studio system, with a focus on remote control of the VESPERS beamline.

SCIENCE STUDIO OVERVIEW

The Science Studio web portal is mostly implemented in Java, and uses server-side web technology common to enterprise applications such as Java Servlets, Java Messaging Service (JMS), Java Database Connectivity (JDBC) and Java Server Pages (JSPs). In addition, many high quality open-source frameworks and libraries have been leveraged to build a highly functional web portal. The Spring [1] framework is used extensively throughout to build very robust and highly configurable servlets using the Model-View-Controller (MVC) architectural pattern. The iBATIS [2] Object-Relational Mapper (ORM) library is used to easily persist objects to a MySQL [3] relational database. The XStream [4] library provides fast object marshalling capabilities in both XML and JSON formats. Security functionality is provided by the JSecurity [5] framework using some custom extensions. Other Java libraries and tools include Apache Log4J [6], Apache Commons [7], Apache Tomcat [8], Apache ActiveMQ [9] and Jetty [10].

Data Model

Science Studio defines and implements a data model to capture the metadata associated with scientific research. Figure 1 is a data object relation diagram for this data model. The objects belonging to the experiment model have been indicated. A primary objective of Science Studio is to allow scientists, and other people, to collaborate; therefore an important part this data model is the *person* object. A *person* represents a user of the system and contains information such as their name, affiliation, email address and mailing address.

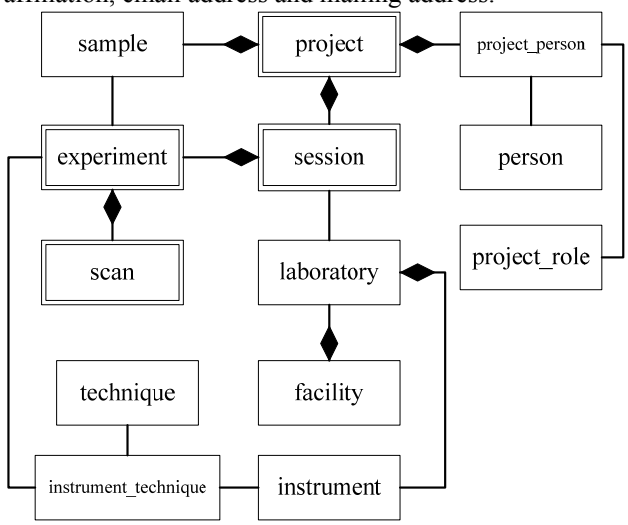


Figure 1: Data object relation diagram for the Science Studio data model, with the experiment model indicated.

Research projects are the foundation of experiment management in Science Studio. For that reason, the *project* object is the top-level organizational element for the hierarchical experiment model. A *project* is composed of *person*, *sample* and *session* objects. The collection of *persons* represents the people collaborating on a project, or simply a project team. A *sample* represents the physical specimen that is the subject of investigation for a project team. A *session* is composed of *experiment* objects and represents the reservation or allocation of resources to the project team for a specified time period. An *experiment* is composed of *scan* objects and references a *sample*, *instrument* and *technique* object. A *laboratory* is composed of *instruments* that are associated by location or function. An *instrument* references *technique* objects and represents a device or resource used to conduct an experiment. A *technique* represents the method or process used by an instrument to produce data. A *scan* represents the actual experimental data produced by an instrument, and contains information about its storage location and file format.

Standard Data Format

The manipulation of experimental data is a requirement for most scientific applications. Science Studio specifies a standard format for experimental data files to facilitate

*This work was funded by CANAIRE under R&D project NEP-01; and performed at the Canadian Light Source which is supported by NSERC, NRC, CIHR and the University of Saskatchewan.
#dylan.maxwell@lightsource.ca

the sharing of data between applications. This format is an extension of the more general purpose Common Data Format (CDF) [11]. The CDF is a self-describing format, with both binary and XML versions, used for efficient storage of scalar and multidimensional data. The standard format defines an overlying structure for the CDF that provides more information about the type of experimental data contained within a data file. Science Studio implements utilities for reading and writing files in the standard format, as well as, a framework for building custom data format converters.

Security

Science Studio provides security features such as authentication, authorization and session management. A web application implements shared services for user authentication such as the login and logout pages. This is indicated in Figure 2 by the Login Servlet. Security session management is handled mostly by JSecurity using a customized servlet filter. A servlet can be easily configured to use this filter, which will only allow access to authenticated users.

Authorization is provided using a project-oriented permission system. The members of a project team are associated with a project role. The project role determines the permissions that each team member has within the project. Currently only two project roles are used: Experimenter and Observer. Experimenters have full access to the project. They are permitted to create, read, edit and delete data objects belonging to the project. Observers are only permitted to read data objects belonging to the project. These permissions also apply to remote access. Experimenters are permitted to control the remote instrument, and Observers are only permitted to view the remote instrument.

Web Portal

Science Studio implements an extensible web portal that gives users a single, consistent entry-point for access to other services. This rich web interface is built using the Ext [12] JavaScript framework. In Figure 2, the server-side of this web application is indicated by the Core Servlet. A primary feature of the web portal is the ability for users to browse the data model. The data model is represented as data trees with *projects* as the roots, and *scans* as the leaves. Users can navigate to data objects using the tree, which will then provide different options based on the data object type. For example, selecting a *scan* allows users to view the experimental data, or selecting a *session* allows them begin remote access. Users can also create, edit and delete data objects, provided they have the required permissions.

VESPERS REMOTE ACCESS

The VESPERS beamline is located at the Canadian Light Source (CLS) synchrotron [13] in Saskatoon Saskatchewan. VESPERS is a microprobe beamline that operates in the energy range of 6 to 30keV using bending magnet radiation. The experimental station is equipped with both a CCD area detector and a four element Silicon Drift Detector (SDD). Together they are capable of multiple complimentary techniques such as X-Ray Diffraction (XRD) and X-Ray Fluorescence (XRF) spectroscopy.

XRD is a common technique used for determining the microcrystalline structure of geological samples. This technique uses the CCD detector to record the diffraction pattern produced by a sample when exposed to a focused x-ray beam. The CCD detector image size is 2084 x 2084 pixels or approximately 8MB. For an area of interest that

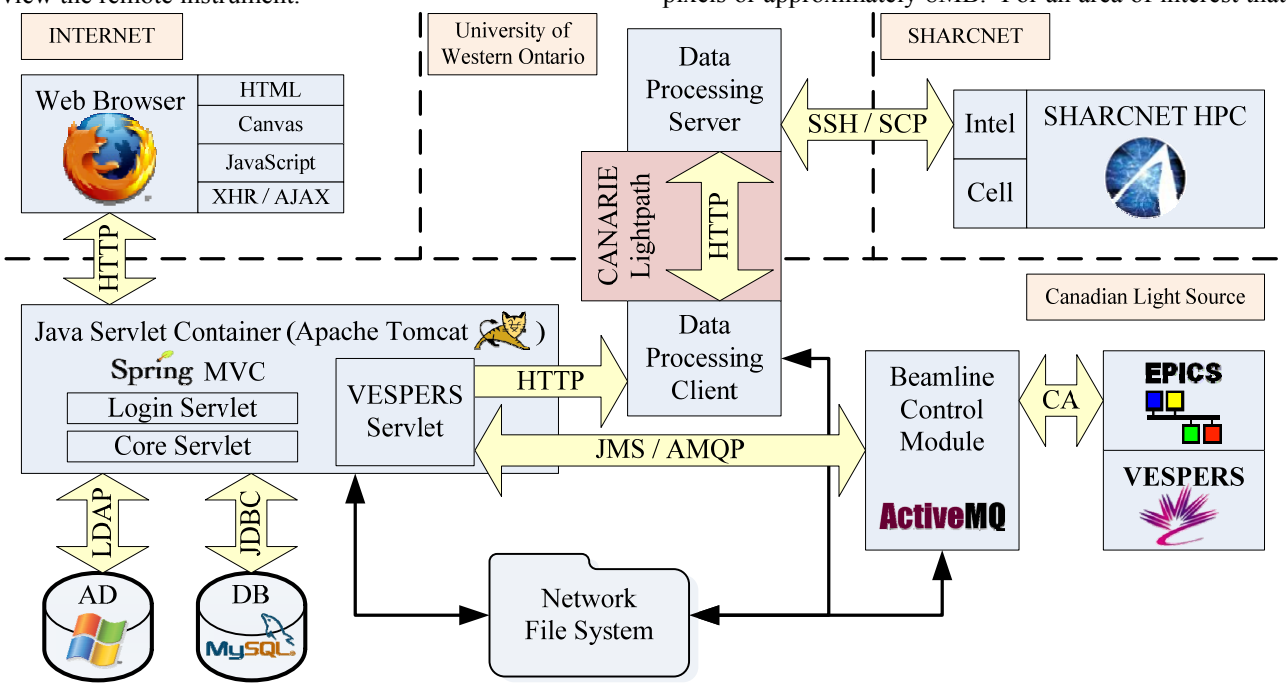


Figure 2: Science Studio architecture for VESPERS beamline remote access and data processing.

is 100 x 100 data points, the total size of the data set is approximately 80GB. A data set of this size requires many hours to process using conventional computers with standard software. However, using the computers at the SHARCHET [14] High Performance Computing (HPC) center, in conjunction with special software, this large data set can be processed in minutes. The SHARCNET HPC center is located at the University of Western Ontario (UWO) in London Ontario.

Science Studio allows users to remotely access the experimental capabilities of the VESPERS beamline, and then to readily utilize the computational capabilities of SHARCNET. Shown in Figure 2 is an architectural diagram of the main components, and their interaction, for the remote access and data processing systems.

EPICS Control System

EPICS [15] is the standard control system at the CLS and is used for control and data acquisition of nearly every device at the facility. The Channel Access (CA) protocol is used to communicate with EPICS over the network.

Beamline Control Module

The Beamline Control Module (BCM) is a Java application which provides a high-level interface to the EPICS control system. The BCM communicates with EPICS using a Java implementation of CA to monitor and change the state of devices on the VESPERS beamline. The BCM provides a device abstraction so that alternate low-level control systems can be used. This is important for use of the BCM outside of the CLS.

Web Application

The VESPERS beamline web application provides a user interface for device control and data acquisition. The Ext JavaScript framework is again used to build a rich interface that uses asynchronous requests to provide frequent (normally once per second) updates to the device information. This web application allows the user to interactively explore the sample, and then define a scan area by simply drawing a rectangle on the sample image. When the user starts a scan they are prompted to enter a name for the scan. The progress of a scan is displayed numerically, as the percentage complete, and graphically, as an animated dot that moves across the scan area. The user can also configure and test both the SDD and CCD detector. The web application also gives access to three video cameras, with pan, tilt and zoom capability, that show various views of the experimental station. Although all members of the project team, who have the Experimenter role, are permitted to control the beamline, only one user at a time is allowed to be in control of the beamline. In Figure 2, the server-side of this web application is represented by the VESPERS Servlet.

Data Processing Service

The raw data collected on the VESPERS beamline must be transferred from CLS to UWO for processing, and the processing results must to be transferred back to CLS for presentation to the users. In order to fully take advantage of the CANARIE Lightpath high-speed connection between the CLS and UWO, the File Transfer Server (FTS) and File Transfer Client (FTC) provide the following features:

- Simultaneous TCP connections.
- Start transferring multiple files with one request.
- Asynchronous client using non-blocking I/O.
- File content compression using *gzip* [16].
- File range transfer over multiple connections.

The File Monitor Service (FMS) provides file system event notification through HTTP. It is difficult to get I/O event notification from the CLS data acquisition system, by which experimental data is collected on the VESPERS beamline. Providing pseudo-realtime processing of experimental data requires that each piece of data be transferred to UWO once it is available on the CLS file system. This service can be deployed on any system with native support for *inotify* [17], and that has the *inotify-java* [18] library installed.

The basic sequence of events for the Data Processing Service is shown in Figure 3. The VESPERS Servlet first sends a request to the FMS to initiate monitoring of a specified directory. The notifications for these events are sent directly to the FTC, which then initiates the transfer of the experimental data files from the FTS.

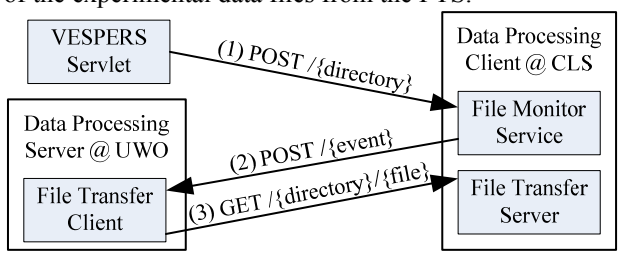


Figure 3: Event sequence for the Data Processing Service.

REFERENCES

[1] Spring <http://www.springsource.org/about>
[2] iBATIS <http://ibatis.apache.org>
[3] MySQL <http://www.mysql.com>
[4] XStream <http://xstream.codehaus.org>
[5] JSecurity (renamed to Apache Shiro) <http://incubator.apache.org/shiro>
[6] Apache Log4J <http://logging.apache.org/log4j>
[7] Apache Commons <http://commons.apache.org>
[8] Apache Tomcat <http://tomcat.apache.org>
[9] Apache ActiveMQ <http://activemq.apache.org>
[10] Jetty <http://jetty.codehaus.org/jetty>
[11] Common Data Format <http://cdf.gsfc.nasa.gov>
[12] Ext JS <http://www.sencha.com/products/js>
[13] Canadian Light Source <http://www.lightsource.ca>
[14] SHARCNET <https://www.sharcnet.ca/my/front>
[15] EPICS <http://www.aps.anl.gov/epics>
[16] gzip <http://www.gzip.org>
[17] inotify <http://inotify.aiken.cz>
[18] inotify-java <https://launchpad.net/inotify-java>

RESEARCH METADATA MANAGEMENT AT THE AUSTRALIAN SYNCHROTRON

Richard. Farnsworth, Alistair Grant, Andrew Rhyder, Australian Synchrotron, Melbourne Australia
Nick Hauser, Bragg Institute ANSTO, Sydney Australia.

Abstract

This paper details the approach the Australian Synchrotron [1] is using, in collaboration with the Australian Neutron Source, run by the Bragg institute, part of ANSTO [2] (Australian Nuclear Science Technology Organisation) called OPAL (Open Pool Australian Light-water Reactor) for some of the data and metadata management issues. It explores the data and user policies, describes the quantity and quality of data and demonstrates the way forward based on both existing and future directions in e-research, network communications, user proposal and material databases, portal technologies and integration techniques. The role of standards for access and metadata creation is also explored. This work is funded by an educational infrastructure grant administered by Australian National Data Services.

DATA POLICY

In order to progress with publicly funded research facilities data and metadata publishing the data policies must be clear. The answer to the questions who owns the data, when can you make it public, what can you do with it should be clear. At the Australian Synchrotron twenty-four months is allowed to the principle investigator to publish publicly. The period is thirty-six for Bragg institute instruments (ANSTO, OPAL). Either facility may choose to process the raw data in order to make is accessible or publishable. There is a growing trend worldwide towards open technical data. There is also a growing trend towards publishing not only a scientific paper, but also the raw data that was used to produce it.. As both the Australian Synchrotron and the Bragg institute are publicly funded, technical data created at either location should be at some point made available to the public. Currently there are some Australian mechanisms for achieving this. One is called the ANDS portal. [3] ANDS stands for the Australian National Data Service.

DATA QUANTITY

The Australian Synchrotron operates nine beamlines producing around two to three Terabytes of experimental data per day across a wide variety of disciplines from protein crystallography, medical, through to the conservation and restoration of cultural objects and works of art. In 2009 over five hundred groups conducted research at the Australian Synchrotron. More are expected this year and the next. The Australian Synchrotron expects to be producing at least eight terabytes per day when the next round of ten beamlines are installed in the coming two to five years. Even if the Australian Synchrotron just keep operating the existing beamlines, there will be a significant increase in data collection because of both the continual improvement in detectors and the overall efficiency or "duty cycle" of the beamlines. The objective of this project to make that data available publicly. The data will be stored in the curated archives immediately; however authorisation for access will be allowed or otherwise depending on when the data can be made available. Much smaller volumes of data are created at the Bragg institute.

This project is seeking to provide services so that researchers and institutions can manage their data. To give them the power of something like "Google" over their data – that is the ability to search, catalogue and access. This promotes the use and re-use of data and so adds to the efficiency of the data generating ability of each facility.

MECAT

The chosen a name for this project is "MeCAT" [4], as a nod to a similar project/product called ICAT. It was a requirement to name the project, rather than the technology being used. The project is to enhance the technology to enable those things aforementioned efficiencies.

COLLABORATION

It is worthwhile noting the collaboration details. The two facilities have decided that if they collaborate and pooled resources between the two similar facilities in Australia, we could effectively get twice the efficiency of the software development dollar in terms of software resources.

It also a major step towards the creating an Australian culture of same software in similar institutions. This leads to the same experiences for researchers. This is becoming increasingly strategically important to both facilities. It would be ideal if every institute used exactly the same software everywhere such that experimenters trained in the use of software one area or instrument could use the same skills in another. This is probably never going to be completely possible, but this project assists by moving toward that philosophical direction. It also helps with the data management, because the software automatically moves the data for those researchers that come from known institutions to their home institutions.

OBJECTIVE OF PROJECT

The objective of this project is to provide services to researchers to manage their experimental data and to provide data search and access to the broader research

community. These services will enable better use and reuse of the data. The ultimate aim is to combine these services into a common environment to allow project teams to interact with the instruments and even allow for a data collaboration between ANSTO and AS

MECAT FOCUS

The Australian Synchrotron intends to focus this project on three of its nine beamlines, because there is a great deal of disparity between all nine - too much for the project to deal with in the first instance. The Australian Synchrotron will be looking at the Soft x-ray spectroscopy, Infrared Microspectroscopy, and Macromolecular Crystallography beamlines. The Macromolecular Crystallography beamline already is using parts of the MECAT project software. The Australian Synchrotron is intending to take data from the experimental end stations, the proposal database, scheduling database and the EPICS control systems. At ANSTO, their scope is a little different. The Bragg institute is considering all of the neutron beam instruments, these instruments are smaller data volume producers, although of no less importance. Instead of EPICS they use a control system which is a local adaptation of the Swiss Spallation Instrument Control System (SICS) a collaboration from the Paul Scherrer Institute (PSI). The actual implementation at the Bragg Institute is nearly identical.

Both institutes will produce an ARCS compatible data repository, ARCS is the Australian Research Collaborative Services. This will then allow a set of standards to harvest that data publicly using metadata.

ARCS

ARCS lets researchers look for data, transfer data and to share material. It uses a concept called the "Data fabric" which is an overloaded term that has only recently been defined more precisely. It's been used like the "cloud", but aligned for research data purposes and has central; data storage, security etc. for facilities across Australia. As time goes by, there will be more and more experimental facilities using the data fabric provided by ARCS. The following description of ARCS [4] is pertinent:

The Australian Research Collaboration Service provides tools and services that enable researchers to operate at the forefront of their fields. It is intended to allow them to securely store large volumes of data for more collaboration. These tools and services also enable the transfer of data for faster analysis and result, to share material for convenience and control and finally to share data securely only with authorised

TARDIS

The MECAT project has chosen to use a particular technology to help collect the data for the databases,. It is called TARDIS, and stands for The Australian Repository for Diffraction ImageS, Ref [5]

TARDIS is a collaborative venture; coming out of the eResearch community. It is Australian and was started at Monash University, Australia [6]. It puts data into the dataset for various communities. It started off as a development for assisting users of the Protein Crystallography beamline. It has been made open source and is used for managing groups of files for a given experiment.

TARDIS available at the website ref [5] and is used for managing a group of files associated with an experiment – as per the following schema.

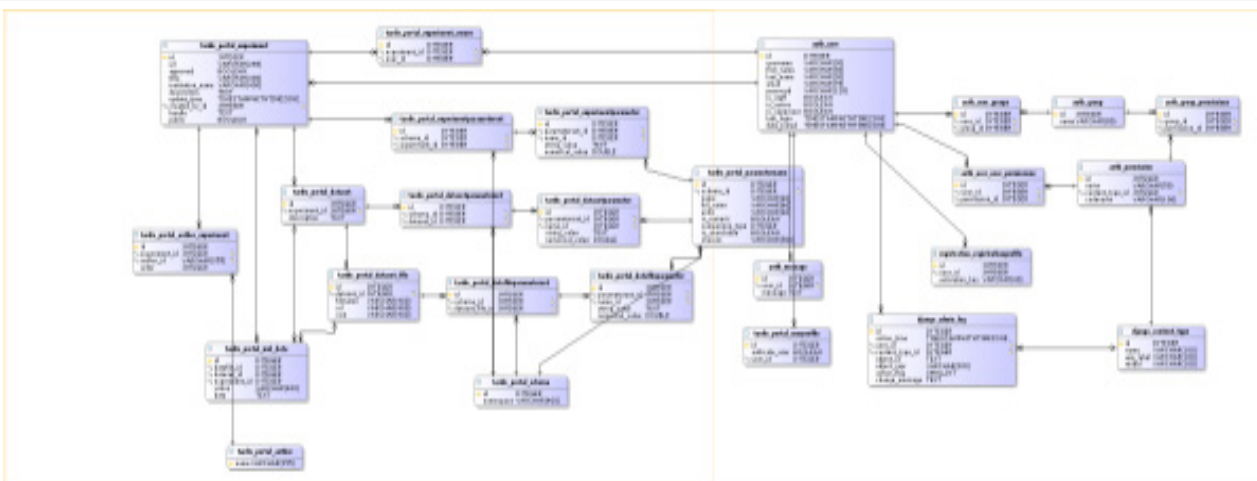


Figure 1: TARDIS Schema showing authorisation and experiment, parameters, database and data files

TARDIS SCHEMA

Figure 1 shows how TARDIS [5] looks like in its full schema. If we remove the authorisation, the complexity reduced significantly. It encompasses an experiment which consists of the "Dublin core" type information – that is title, author date etc. Then there are the soft parameters stored against that. Those parameters will generally be unique to the instrument and science being used for those experiments.

Now examine at the datasets themselves, there may be many datasets associated with a given experiment. This is the way that works for the two facilities in question and similar intuitions. Finally there are the files themselves and where they are located.
In summary, the schema consists of the experiment, the experimental data, datasets and files.

CONCLUSION

MeCAT is a joint project between the Australian Synchrotron and ANSTO to improve Metadata management and publication at the facilities. It is using and extending open source tools called TARDIS and will offer Australian Scientists greater capabilities to share and reuse data.

REFERENCES

[1] http://www.synchrotron.org.au/
[2] http://www.ansto.gov.au/
[3] http://ands.org.au/
[4] http://mecatproj.wordpress.com/
[5] http://tardis.edu.au
[6] http://www.monash.edu.au/

DIAMOND'S TRANSITION FROM VME TO FIELDBUS BASED DISTRIBUTED CONTROL

I.J.Gillingham, S.C.Lay, R.Mercado, P.Hamadyk, M.R.Pearson, T.M.Cobb, M.T.Heron, N.P.Rees
Diamond Light Source, Oxfordshire, UK

Abstract

The interface layers of Diamond's accelerator and photon beamline control systems have predominantly been implemented as VME-based systems. Forthcoming control systems, for new photon beamlines, have requirements necessitating a divergence from Diamond's adopted design patterns, including a reduction in available rack space, and we also need to consider the management of hardware obsolescence. To address these issues, a new standard based on PCs and Ethernet field buses to the instrumentation has been defined. This paper will present the new design, how the design transition is being effected and the key benefits to Diamond.

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[1]. 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 proposed, and subject to funding, detailed design and construction of these 10 beamlines will commence from 2011.

In planning for the next phase of photon beamlines, it was timely to consider the control system architecture applied to future beamlines, associated front ends and experimental stations.

EXISTING CONTROL SYSTEM ARCHITECTURE

Accelerator and beamline control systems use a consistent approach to interface to the hardware, with most equipment interfaced through embedded VME systems. To support the interface requirements of the equipment, a range of I/O modules based on Industrial Pack (IP) modules (ADC, DAC, Serial, DIO) and VME modules (IP carrier, motion, scalar and timing) is used. The field signals are interfaced via either transition modules or front-panel connections. A VME microprocessor (MVME5500) runs VxWorks and EPICS to serve up the control information to client applications. There are in excess of 250 VME-based systems running as part of Diamond's control system[2]. In addition, the electron BPMs run EPICS IOCs directly on the Libera beam processing hardware, and soft IOCs running under Linux on PC hardware concentrate and process data or interface to network attached devices over manufacturer-specific protocols. One anomaly to this approach has been video cameras which have been interfaced to the VME IOC using Firewire and a PMC Firewire adapter located on the VME processor board.

REASON FOR CHANGE

The existing control system architecture has served well for the existing accelerators and beamlines; however it was defined nearly ten years ago, so in the context of the next phase of beamlines the opportunity to reconsider the standards is being taken. In doing so, it is clear that not all the hardware capability of VME is required for beamline control; neither is the use of a hard real-time operating system such as VxWorks. It is also apparent that most I/O functionality required for control of beamline equipment can now be realised through Ethernet-attached I/O. There is also now good infrastructure for developing and managing Linux based EPICS IOCs on a PC architecture.

OUTLINE REQUIREMENTS FOR PHOTON BEAMLINES

In considering the requirements for photon beamline control the following technical systems are identified:

- Motion control
- Vacuum instrumentation and other serial devices
- Video cameras
- Analogue and digital signals
- Programmable logic controllers
- Timing signals

The interface from the IOC to the equipment should make use of the installed network cabling, thereby reducing I/O-specific cabling and giving flexibility in reconfiguration and addition of equipment without the need to pull new cables.

There should be greater partitioning of the IOC functionality by technical area, e.g. motion, camera and vacuum, by running a greater number of EPICS IOC instances, either as separate processes on one Linux system or as single processes, each on a virtualised Linux system. This would minimise the disturbance to beamline operation when making changes that necessitate restarting an IOC.

The I/O associated with the control system should be located close to the equipment being interfaced; i.e. for signals located in experimental and optics hutches, the I/O modules should be co-located in these areas. However, this is constrained by the possibility of radiation-induced damage to I/O in the optics hutches of high energy

(~100keV) beamlines and by the space available in the some beamline hutches.

NEW SOLUTION

Each IOC will run on a 1U Linux PC located within the beamline instrumentation area. This is not regarded as a "soft IOC" as the hardware is connected directly to it. It will probably have several physically separate network connections to support the different systems, so that equipment with limited network stack and CPU capability such as PLCs is not affected by high-data-rate devices such as cameras operating in multicast mode. The structure is shown in Figure 1.

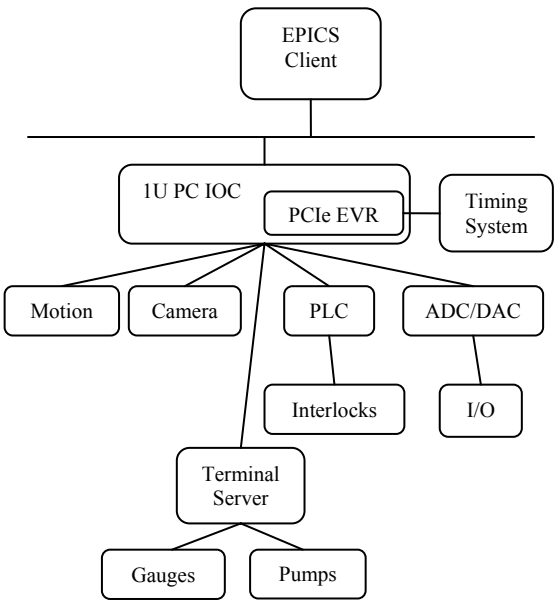


Figure 1: Hardware Architecture

Motion Control

For motion control, a standard based on the Delta Tau Geobrick LV Ethernet-based motor controller is used[3]. This provides 8 axes of motion control and comes complete with amplifiers in a 4U rack-mount box. The existing EPICS motor record software, already in use with older VME hardware, was modified to be compatible with this controller. This was realised by adding an ASYN interpose layer, which provides support for the Delta Tau Ethernet TCP/IP packet structure, and so avoids making changes to the existing PMAC motor controller ASYN driver.

Vacuum Instrumentation and other Serial Devices

Vacuum instrumentation (Gauges and Pump Controllers) and other serial devices will be interfaced through RS232, RS422 or RS485 serial connections. These will connect to a terminal server located in the instrumentation rack and the terminal server via Ethernet to the IOC. On the IOC most serial devices are handled by the EPICS Stream Device module communicating to the serial interface over virtual serial connections to the terminal server.

Cameras on Ethernet

New diagnostic applications will use a range of GigE cameras from AVT (formerly Prosilica).

A video server will run an EPICS IOC using areaDetector[4] to control, process and store images from up to 10 cameras and ffmpegServer[5] for visualisation. AreaDetector is a modular system of EPICS drivers and plug-ins that can be "rewired" at run time, allowing a flexible image processing chain to be set up. Plug-ins for controlling the camera, providing statistics on the images that are produced, filtering them and writing them to disk are included with areaDetector. FfmpegServer is a Diamond-produced plug-in that compresses a stream of images to mjpg and serves them over http.

Programmable Logic Controllers

Interlocking and protection of equipment is realised in Omron CJ1 PLCs. These will be interfaced to the IOC using Ethernet and the FINS[6] protocol over UDP. An EPICS driver has been developed and provides direct read/write access to each PLC's I/O register and memory areas.

The Omron CJ PLC will optionally use remote I/O modules called SmartSlice[7] which will be located in the beamline optics and experiment hutches. The SmartSlice remote I/O comprises a Communications Unit and a number of I/O Units providing digital I/O, analogue I/O, temperature, counter and positioning interfaces.

The SmartSlice I/O Units communicate with the host PLC over a private Ethernet connection running the PROFINET protocol**.** PROFINET[8] provides flexibility so that it is simple to configure additional I/O modules. The interface is realised over standard Ethernet connections.

ADCs DACs and DIO

To interface ADCs, DACs and digital I/O, a range of I/O modules from Beckhoff Automation (Verl, Germany) has been selected. These use EtherCAT[9], an industrial Ethernet-based fieldbus system. This I/O will be used for all non-interlocking type applications, and provides lower latency from the plant to the IOC than the PLC solution. It further minimises the number of I/O points in the PLC-based interlocking system and so minimises the need for changes to the PLC which necessitate revalidation of the interlock logic.

The EtherCAT protocol provides low-latency data transfer from the I/O modules into the host computer. It operates on the principle of a master that communicates with slaves using EtherCAT telegrams that are passed around each node and back to the master. The EtherCAT master uses standard Ethernet controller hardware and a software implementation of the EtherCAT functionality, whilst the slaves use a custom slave controller.

The custom interface implements a Fieldbus Memory Management Unit (FMMU), which allows the mapping of logical addresses in the telegram to physical ones within the slave. This processing occurs on the fly as one slave passes the telegram through to the next slave, introducing a delay of a few nanoseconds. Slaves also automatically close a communication ring when the outgoing Ethernet link (downstream section) is not connected, by returning the telegram to the master back through the chain of slaves.

The telegram structure 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 address devices that may have a payload of only a few bytes, such as digital I/O devices that are typical in industrial automation.

Although the protocol will operate with other Ethernet-based services and protocols on the same physical network, the proposed Diamond Remote I/O solution will adopt strict segregation of the EtherCAT bus.

Because we are using Linux, the hardware supported is limited to Realtek and Intel cards, plus a 'generic' interface.

Timing Signals

The Diamond timing system is applied across the accelerators and beamlines[10]. On the beamlines it is used to decode orbit and bunch clocks to enable synchronisation of experiments to the stored beam structure. It provides gating signals which at injection, during top-up operation, are used by beamline detectors to mask out the stored beam disturbance. The timing system also provides time stamps for EPICS record processing. To support this functionality in the new architecture, it is envisaged that a PCIe version of the Event Receiver module will be developed. This will make the time stamp information available in the PC-based IOC and will bring out the decoded signals on a 1U interface panel.

SUMMARY OF PROGRESS TO DATE

Ethernet-based motion control subsystems are already implemented and deployed on a number of beamlines connected to both PC and VME IOCs. They have proved to provide effective control of stepper and servo motor systems, e.g. monochromators, slits, mirrors etc. Remote diagnostics and configuration are also proving to be very valuable.

Similarly interfacing a range of instruments over terminal servers is also actively being used and makes use of already developed Streams support modules.

The FINS interface to the Omron PLC has been implemented and deployed to integrate a single PLC controlling LN2 distribution. The design of standard remote I/O modules has also been undertaken. Given the risk of possible radiation damage, SmartSlice remote I/O units have been in soak-test for the past two months in one of Diamond's optics hutches. The implementation of SmartSlice systems is being planned for forthcoming beamline control and front-end equipment protection systems.

The EtherCAT based remote I/O has been through initial evaluation and testing with a Linux x86 PC as a host. Initial tests have been performed using an Intel E1000 controller on a standard RHEL5 dual-core Intel Pentium 4 Xeon PC. A user-space polling process, fully using one of the two available cores, was able to reproduce a pulse read from an ADC and to drive a digital output with a delay of 200 microseconds. Further effort is planned to develop EPICS device support for the various EtherCAT I/O modules to be used on Diamond.

REFERENCES

[1] R.P.Walker, “Commissioning and Status of The Diamond Storage Ring”, APAC 2007, Indore, India.
[2] M.T.Heron et.al., “Implementation, Commissioning and Current Status of the Diamond Light Source Control System”, ICALEPCS 2007, Knoxville, 2007.
[3] N.P.Rees et.al., “Development of Photon Beamline and Motion Control Software at Diamond Light Source”, ICALEPCS 2007, Knoxville, 2007.
[4] http://cars9.uchicago.edu/software/epics/areaDetector.html
[5] http://controls.diamond.ac.uk/downloads/support/ffmpegServer/
[6] OMRON FINS Commands Reference manual (W227-E1-2)
[7] OMRON Slice I/O Units Operation Manual (W455-E1-06)
[8] OMRON GRT1-PNT PROFINET I/O Communication Unit Operation Manual (W13E-EN-01)
[9] ETG. EtherCAT - the Ethernet fieldbus. http://www.ethercat.org/pdf/ethercat_e.pdf .
[10] Y. Chernousko, P. Hamadyk, M. T. Heron, “Review of the Diamond Light Source Timing System”, RUPAC 2010, Protvino, Russia, 2010

A DISCRETE HYSTERESIS MODEL FOR PIEZOELECTRIC ACTUATOR AND ITS PARAMETER IDENTIFICATION

Y. Cao and X. B. Chen#
Department of Mechanical Engineering
University of Saskatchewan, Canada

Abstract

Hysteresis is an important nonlinear effect exhibited by piezoelectric actuators (PEA) and its modelling has been drawing considerable attention. This paper presents the development of a novel discrete model based on the concept of auto-regressive moving average (ARMA) for the piezoelectric-actuator hysteresis, and its parameter identification method as well. Experiments were carried out to verify the effectiveness of the developed model. The result obtained shows that the developed model can well represent the hysteresis of the PEA.

INTRODUCTION

Piezoelectric actuators (PEA) have been widely used in nanopositioning applications, such as AFM , STM , DVD disc reading and writing [1], diamond lathe machine [2], lithography, X-ray imaging [3]. However, the performance of a PEA can be significantly degraded by its hysteresis. Hysteresis is a memory effect of piezoelectric actuators and, as a result, the hysteresis exhibited at an given time instant depends on not only the input at the present time but also the operational history of the system considered. In order to develop control schemes on PEA, modelling of PEA has been drawing considerable attention and several models have been resulted to describe the hysteresis effect, such as Preisach model [4], the ferromagnetic material model [5] and the nonlinear auto-regressive moving average model with exogenous input (NARMAX model) [6]. However, most of the models developed in literatures are continuous and the model-based controller design is proceeded in the continuous time domain. With the advance of computer technology nowadays, controllers are mostly implemented digitally. Note that not all the continuous controllers can work on the sampled digital system as desired since the discrete sampling can sometimes make the continuous system unstable. Therefore, it is advantage to develop a discrete hysteresis model of PEA for its digital controller design. Unfortunately, little work about the discrete hysteresis model or the digital controller design for PEA has been found yet. In this paper, the ferromagnetic material hysteresis model is discreted and, by combining it with the concept of auto-regressive moving average (ARMA), a novel model is developed to represent the hysteresis of PEA. Specifically, the next section of this paper is the introduction to the discrete ARMA-based hysteresis model, which is followed by the experimental identification and verification results by using the discrete ARMA-based hysteresis model as compared to the general discrete form of hysteresis model [7]. The last section gives the conclusions of the paper and future work.

xbc719@mail.usask.ca

DISCRETE ARMA-BASED HYSTERESIS MODEL

The ferromagnetic material hysteresis model introduced by Adriaens and Koning [5] is illustrated in the following:

$$\dot{y} = \alpha|\dot{x}|[f(x) - y] + \dot{x}g(x) \tag{1}$$

where x is the input of the hysteresis and y is the output, $f(x)$ and $g(x)$ are functions of x with which you can "shape" the hysteresis loop. It has been experimentally verified that this differential equation is also suitable for describing electric hysteresis such as PEA. In theory, PEA shows the length saturation. In practise, however, the displacement of the PEA stays far away from saturation. Therefore, chose $f(x) = ax/\alpha$ and $g(x) = b$ as the shape function, Equation (1) can be rewritten as:

$$\dot{y} = |\dot{x}|(ax + cy) + b\dot{x} \tag{2}$$

where $c = -1/\alpha$. [7] applied the difference equation to discrete Equation (1) as follows:

$$\begin{aligned} y(k+1) - y(k) = {} & |x(k+1) - x(k)|[ax(k) + cy(k)] \\ & + b[x(k+1) - x(k)] \end{aligned} \tag{3}$$

This paper discrete Equation (1) by integral.

Discrete form of the ferromagnetic material hysteresis model

When the input signal is monotonically increasing, $\dot{x} > 0$, take integral on both side of equation (2) in one sampling interval, one can derive:

$$\int_{kT}^{(k+1)T} \dot{y}dt = a\int_{kT}^{(k+1)T} \dot{x}xdt + c\int_{kT}^{(k+1)T} \dot{x}ydt + b\int_{kT}^{(k+1)T} \dot{x}dt \tag{4}$$

where T is the sampling interval.

Equation (4) leads to:

$$\begin{aligned} y(k+1) - y(k) = {} & \frac{1}{2}a[x^2(k+1) - x^2(k)] + c\int_{x(k)}^{x(k+1)} ydx \\ & + b[x(k+1) - x(k)] \end{aligned} \tag{5}$$

which is the discrete form of the first order hysteresis differential equation (2).

Using trapezoid equation to estimate the integral term, Equation (5) yields:

$$y(k+1) = a\frac{\alpha(k+1)}{2 - c\beta(k+1)} + b\frac{2\beta(k+1)}{2 - c\beta(k+1)}$$

$$+\frac{2+c\beta(k+1)}{2-c\beta(k+1)}y(k) \quad (6)$$

where $\alpha(k+1)=x^2(k+1)-x^2(k)$,

$\beta(k+1)=x(k+1)-x(k)$.

Given the zero initial condition that $y(1)=0$,one can derive:

$$y(2)=a\frac{\alpha(2)}{2-c\beta(2)}+b\frac{2\beta(2)}{2-c\beta(2)}$$

$$y(3)=a[\frac{\alpha(3)}{2-c\beta(3)}+\frac{2+c\beta(3)}{2-c\beta(3)}\cdot\frac{\alpha(2)}{2-c\beta(2)}]$$

$$+b[\frac{2\beta(3)}{2-c\beta(3)}+\frac{2+c\beta(3)}{2-c\beta(3)}\cdot\frac{2\beta(2)}{2-c\beta(2)}]\cdots$$

Therefore, the output y can always be represented as a function of input x by recursion:

$$y(k+1)=ay_1(k+1)+by_2(k+1) \quad (7)$$

where

$$y_1(k+1)=\frac{\alpha(k+1)}{2-c\beta(k+1)}+\frac{2+c\beta(k+1)}{2-c\beta(k+1)}y_1(k) \quad (8)$$

$$y_2(k+1)=\frac{2\beta(k+1)}{2-c\beta(k+1)}+\frac{2+c\beta(k+1)}{2-c\beta(k+1)}y_2(k) \quad (9)$$

When the input signal is monotonically decreasing, $\dot{x}<0$, repeating the above process, one can derive:

$$y(k+1)=ay_1(k+1)+by_2(k+1) \quad (10)$$

where

$$y_1(k+1)=\frac{-\alpha(k+1)}{2+c\beta(k+1)}+\frac{2-c\beta(k+1)}{2+c\beta(k+1)}y_1(k) \quad (11)$$

$$y_2(k+1)=\frac{2\beta(k+1)}{2+c\beta(k+1)}+\frac{2-c\beta(k+1)}{2+c\beta(k+1)}y_2(k) \quad (12)$$

In order to verify the effectiveness of the discrete form of the first order hysteresis differential equation, a SIMULINK model was built to generate the simulation data. Parameters a, b, c are chosen to be 0.0064, -0.0378, 0.1144 such that the output displacement of the SIMULINK model can fit the measured displacement. Meanwhile, another group of output data was generated by Equations (3) and (7)-(12). Table 1 shows the comparison of the discrete error using Equation (3) and Equation (7)-(12). The input is a sinusoidal signal whose frequency is set to be 1Hz ~300Hz with 70V magnitude. From the result, it can be concluded that the discrete hysteresis Equation (7)-(12) is more accurate in describing hysteresis than the general Equation (3).

Table 1: Discrete error by using different methods

Input Frequency (Hz)	**50**	**100**	**200**	**300**
By using Equation (7)-(12)	0.0167	0.0335	0.0671	0.1010
By using Equation (3)	0.0171	0.0341	0.0681	0.1022

Discrete ARMA-based hysteresis model

The general ARMA model has the form as follows:

$$z(t)=\sum_{i=1}^{N_z}a_i z(t-i)+\sum_{i=0}^{N_y}b_i y(t-i) \quad (13)$$

X.B. Chen, Q.S. Zhang et. al. [8] have made a conclusion that the second order system can be used to approximately represent the dynamics of the piezoelectric stage if the mass ratio between the stage and the actuator increases. Thus, using a second order ARMA model, one can derive:

$$z(t)=a_1z(t-1)+a_2z(t-2)+b_0y(t)+b_1y(t-1)+b_2y(t-2) \quad (14)$$

Substitute the discrete hysteresis Equation (7) and (10) into Equation (14), the discrete ARMA based hysteresis model will be derived as:

$$z(t)=a_1z(t-1)+a_2z(t-2)+b_0'y_1(t)+b_0''y_2(t)$$

$$+b_1'y_1(t-1)+b_1''y_2(t-1)+b_2'y_1(t-2)+b_2''y_2(t-2) \quad (15)$$

It will be used to describe the rate-dependent performance of a piezoelectric actuator later.

PARAMETER IDENTIFICATION AND EXPERIMENTS

Experiments are implemented on a PEA (P-753, Physik Instrumente). The actuator can generate displacement in a range of 15 μm with a resolution of 0.5 nm. For displacement measurements, a capacitive displacement sensor of the P-753 PEA is used. It is a built in sensor with a resolution of 1nm. Both the actuator and the sensor are connected to a host computer via an I/O board (PCI-DAS1602/16, Measurement Computing Corporation) and controlled by SIMULINK programs. All measured displacements used in this study were measured with a sampling interval of 0.05 ms. The unit of the measured displacements is μm. The mass ration of the stage and the PEA is 49.8 which indicates that the dynamics of the piezoelectric driven stage can be regarded as a second order system approximately according to our previous study [8].

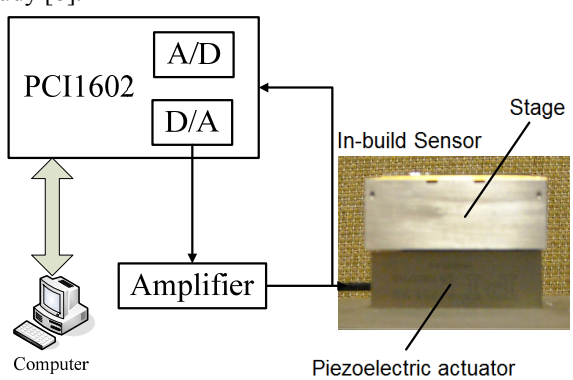


Figure 1: Piezoelectric driven stage

Online estimation method is applied to identify the parameters in the model by giving a bunch of sinusoidal inputs with frequency varying from 10Hz to 200Hz and amplitude being 70V. Table 2 shows the parameter

identification results for discrete ARMA-based hysteresis model. The parameter c is identified to be -0.0305 by using Box-Kanemasu method. The initial value of the hysteresis operator parameters a, b, c are still identified from the 1Hz 70V sinusoidal input data using LS method.

Table 2: Discrete error by using different methods

Parameters	a_1	a_2	b_0'	b_0''
Value	1.6531	-0.676	-0.00276	-0.0732
Parameters	b_1'	b_1''	b_2'	b_2''
Value	0.0064	0.174	-0.00353	-0.0976

Another two types of inputs is applied to the piezoelectric actuator and the corresponding output is measured for model verification. One is the piecewise continuous combination of different amplitude sinusoidal inputs with the same frequency. The other one is the superposition of four sinusoidal inputs with different frequency, amplitude and phase delay.

Table 3 shows the estimation error according to discrete ARMA-based hysteresis model when applied a piecewise continuous combination of different amplitude sinusoidal inputs. The frequency varies from 10Hz to 400Hz. In order to show the effectiveness of the discrete method developed in this paper, the estimation error is compared with the general discrete form by using difference equation referred in [7].

Table 3: Estimation error for the Piecewise continuous combination of different amplitude sinusoidal inputs

Frequency (Hz)	**10**	**50**	**200**	**400**
Discrete ARMA-based hysteresis model	0.0943	0.0989	0.1112	0.1603
General discrete form of hysteresis	0.0946	0.0996	0.1128	0.1627

Compare with the model error corresponding to the same type of input data, it can be concluded that as the input frequency increases, the estimation error increases for both discrete methods. Meanwhile, the discrete ARMA-based hysteresis model has a lower estimation error than the general discrete model shown in [7], especially at high frequencies.

CONCLUSIONS

This paper presents the development of a novel discrete ARMA based hysteresis model to describe the hysteresis of PEA. Online estimation method was applied to identify the model parameters. In order to illustrate the effectiveness of the ARMA-based hysteresis model, experiments are carried out and the results are compared with the general discrete model (3). It shows that the discrete ARMA-based hysteresis model can better predict the hysteresis of PEA. However, the model shows a larger estimation error in high frequency application than in low frequency application due to the estimation of the integral term in the discrete hysteresis equation. Using a higher order polynomial equation to estimate the integral term maybe helpful to improve the discretization. Therefore, a piece of the future work will be on the use of a high order polynomial equation to estimate the integral term to improve the behaviour of the discrete ARMA-based hysteresis model. Moreover, the discrete control scheme will be developed to insure the stability of the digital control system for the PEA.

ACKNOWLEDGEMENT

The support to the present study from the China Scholarship Council (CSC) and the Natural Sciences and Engineering Research Council (NSERC) of Canada is acknowledged.

REFERENCES

[1] Santosh Devasia and S. O. Reza Moheimani, A Survey of Control Issues in Nano-positioning, IEEE Transactions on Control System Technology, Vol.15, No.5, Sep.2007

[2] H. J. Zhang,a_ S. J. Chen, and M. Zhou, Fast tool servo control for diamond-cutting microstructured optical components, J. Vac. Sci. Technol. B 27,,3…, May/Jun 2009

[3] H. Zhang, D. Chapman, Z. Zhong, C. Parham, M. Gupta, Crystal tilt error and its correction in diffraction enhanced imaging system, Nuclear Instrument and Methods in Physics Research A 572 (2007) 961-970

[4] I. D. Mayergoyz, Mathematical Models of Hysteresis, Physical Review Letters, Vol.56, No. 15, April 14th, 1986

[5] Han J.M.T.A. Adriaens, Willem L. de Koning and Reinder Banning, Modeling piezoelectric actuators, IEEE/ASME transactions on mechatronics, VOL.5, No.4, Dec. 2000

[6] Liang Deng, Yonghong Tan, Modeling hysteresis in piezoelectric acuators using NARMAX models, Sensors and actuators, A149, 2009, P106-112

[7] Jozef Vörös, Modeling and Identification of Hysteresis using Special Forms of the Coleman-Hodgdon Model, Journal of Electric Engineering, Vol.60, No.2, 2009, 100-105

[8] X. B. Chen, Q. S. Zhang, D. Kang, and W. J. Zhang, On the dynamics of piezoelectric positioning systems, Review of scientific instrument 79. 116101, 2008

AUTOMATION OF THE MACROMOLECULAR CRYSTALLOGRAPHY BEAMLINES AT THE CANADIAN LIGHT SOURCE

M.N. Fodje*, R. Berg, G. Black, P. Grochulski, K. Janzen, Canadian Light Source, 101 Perimeter Road, Saskatoon, SK, Canada S7N 0XN.

Abstract

The Canadian Macromolecular Crystallography Facility (CMCF) is a suite of two beamlines 08ID-1 and 08B1-1. Beamline 08ID-1, is an undulator beamline for studying small crystals and crystals with large unit cells, while beamline 08B1-1 is a bending-magnet beamline for high-throughput macromolecular crystallography with a high level of automation. The primary method of access to CMCF 08B1-1 will be remote, in what is commonly referred to in the field as "Mail-in" crystallography. We are developing a software system for automating both beamlines, with modules for beamline control, experiment control, data analysis, information management, and graphical user interaction. The system is developed using the Python programming language and makes use of popular open-source frameworks such as Twisted, Django and GTK+. Once completed, the system will allow automation of the macromolecular crystallography experiment from experiment setup to data analysis, thereby increasing the efficiency of the CMCF beamlines and reducing the need for user travel to the synchrotron.

BACKGROUND

The growing impact of macromolecular structural analysis to pharmaceutical, academic and industrial research has resulted in a growing demand for access to protein crystallography beamlines. This demand is reflected not only in the number of samples available for analysis, but also in the increased number of scientists from different fields now using structural information in their research. As a result, many more users with less crystallographic training are demanding access to macromolecular crystallography (MX) beamlines at synchrotron facilities. Fortunately, the MX experiment is highly amenable to automation [1]. It is not surprising therefore that there are many on-going efforts by various synchrotron facilities to provide highly automated MX beamlines to the community of users [1-3].

The synchrotron MX experiment can be broken down into distinct steps (see Table. 1). These include sample preparation, beamline setup, sample mounting, sample alignment, sample characterisation, data acquisition and data processing. The details of each step may vary based on the specific sample being examined and type of experiment desired With the exception of the first step, which is usually carried out by experimenters at their home laboratories, the remaining steps can be automated to a very high degree. It is therefore possible in principle to build a fully automated beamline where experimenters simply prepare and send their samples to the beamline, data is automatically acquired, and experimenters are never needed on-site.

Table 1: Steps involved in an MX experiment

Step	Description
Sample Preparation	Samples are frozen in cryogen at the home laboratory and couriered to the synchrotron by experimenters
Beamline Setup	The beamline is configured and optimized
Sample Mounting	The sample is mounted on a Goniometer
Sample Alignment	The sample on the Goniometer is positioned such that the sample rotates within the X-ray beam, for data acquisition
Sample Characterisation	Initial data frames are collected and processed to obtain improved parameters to be used for data acquisition
Data Acquisition	Data frames are collected
Data processing	Data frames are integrated and reduced to reflection files for further analysis and structure determination by the users

Automation of an MX beamline requires tight integration of various hardware and software components. In addition to the beamline hardware required for delivery of a high-quality and stable beam at the sample position, robotic sample mounting devices and computer hardware for data processing are also required. The software system is a central component of every automated beamline and great care has to be taken to ensure that it is reliable and enables the acquisition of the best possible data. Here, we describe the architecture and implementation of the software for automation of the 08B1-1 and eventually the 08ID-1 beamlines at the Canadian Light Source (CLS).

SOFTWARE ARCHITECTURE

The software system being developed for automation of the CMCF Beamlines is a modular system, layered above the low-level beamline instrument control system which is based on the Experimental Physics and Industrial Control System (EPICS). The main system modules are the Experiment Management Module (EMM), the Beamline Control Module (BCM), the Data Processing Module (DPM), and the Information Management Module (LIMS) (see Fig. 1).

* Email: Michel.fodje@lightsource.ca

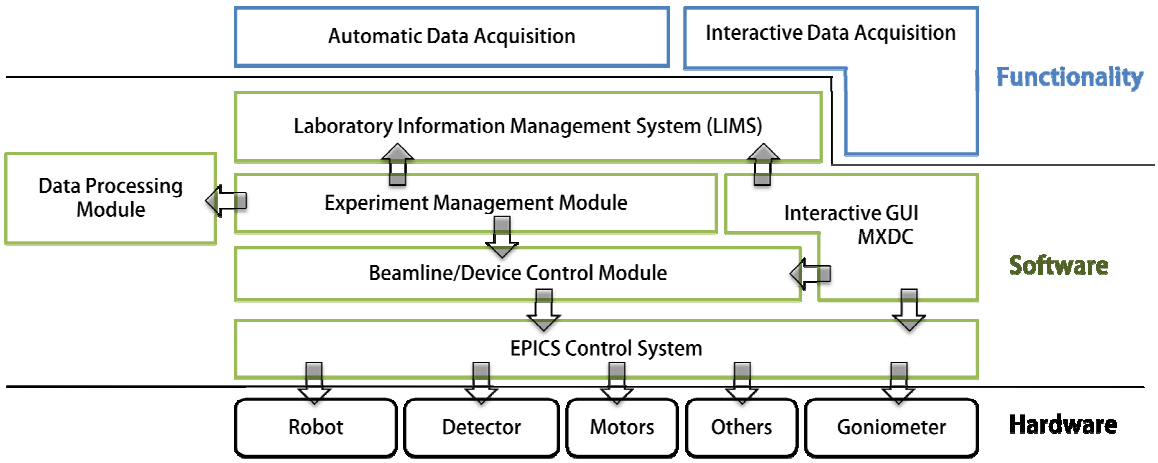


Figure 1: CMCF Software Architecture for automation, showing the various software modules, cross communication between the modules, and provided functionality.

Information Management Module

The Information Management Module, also known as the Laboratory Information Management System (LIMS) is responsible for the storage of information about samples, sample shipments, experiment requests, experiment results and data sets. This module provides a web-based interface for users to submit sample information and review experiment results, and also a web-based interface for beamline staff to manage beamline sessions. It also provides crystal information and experimental requests to the EMM.

Beamline Control Module

The Beamline Control Module (BCM) is responsible for directly controlling the beamline hardware. This module is a high-level module, which must be distinguished from the EPICS based beamline control system. The BCM is an integrated unit which controls the beamline hardware through the EPICS Channel Access protocol, in order to carry out the following functions automatically:

- Beamline configuration.
- Beam optimization.
- Sample mounting and dismounting.
- Sample alignment.
- Data acquisition.

Data Processing Module

The Data Processing Module (DPM) is in charge of integrating and reducing collected data frames into crystallographic reflection files. Specifically the DPM carries out the following functions:

- Scoring of samples to assess quality and suitability for data acquisition.
- Determination of sample parameters and an optimum strategy for data acquisition.
- Integration and reduction of diffraction images into reflection files.
- Data conversion into user-friendly formats for further processing.

Experiment Management Module

The experiment management module (EMM) is at the top of the of the beamline software hierarchy. The role of this module is to substitute the experimenter in carrying out scheduling and coordination of the steps involved in the MX experiment. As a result, the functioning of this module is experiment-centric rather than beamline-centric. This module delegates tasks to the other modules based on experiment information received from the LIMS. For example, sample mounting is delegated to the BCM, while sample characterisation is delegated to the DPM.

IMPLEMENTATION

The beamline software system is implemented in the Python programming language. Python is an interpreted, object oriented, high-level programming language, suited for rapid application development [4]. The availability of several high-quality Python-based frameworks for graphical interface development, web-application development and network server-client development made Python an obvious choice for the implementation language.

The LIMS is implemented using the Django web-application framework [5]. External interfaces used for exchanging information with other modules such as the Experiment Management Module, are implemented as JSON-RPC (JavaScript Object Notation Remote Procedure Call) interfaces [6].

The other modules, the BCM, EMM and DPM, are implemented using a combination of the Twisted Framework, and the Glib/GObject system. The Glib/GObject system, a part of the GTK+ toolkit [7] is a low level library which provides interfaces for event-

driven programming together with a dynamic object system. It is widely used for graphical interface development but can be used for non-graphical applications. The BCM uses GObject extensively to achieve an event driven architecture, where changes in the control system are propagated asynchronously to the rest of the system as they occur. For example, data acquisition is automatically paused when the synchrotron beam is no longer available. The GTK+ toolkit is also used for all interactive graphical interfaces such as the MX Data Collector (MXDC) (used for interactive data acquisition at the beamlines).

The Twisted Framework is a highly flexible, secure and stable networking engine written in Python with interfaces for inter-process communication, asynchronous programming, and web-application development [8]. The Perspective Broker modules are used for all communication between the BCM, DPM and EMM. This includes remote method calls and transmission of data objects. Twisted is also used to provide remote administrative python consoles for live debugging of the modules.

The BCM, DPM and EMM also make use of the Multicast DNS (Domain Name System) service discovery protocols to publish and discover configuration parameters of available services to which other modules may connect. For example, the EMM automatically determines at run-time the host address and port of the BCM, and DPM services which can be running on any machine within the local network. In addition, services can easily be migrated to different hosts and other modules will automatically be notified to reconnect at run-time without manual reconfiguration.

The software system relies on existing established software packages from the crystallographic community for specialized functions. Specifically, the BCM makes use of the XREC package [9] for automatic alignment of samples. Furthermore, the DPM makes use of the XDS [10], CCP4 [11] and BEST [12] software packages.

CONCLUSION

The CMCF Software system will enable remote access to the facility in what is usually referred to as "Mail-in" crystallography, with a high level of automation. Users will prepare their samples at their home labs and ship the samples to the CLS. Using information provided by the users through the web-based LIMS, data will be automatically collected by the EMM delegating to the BCM and DPM as appropriate. After review of the results by beamline staff, the results will be made available to users through the LIMS. When fully functional, this mode of operation will increase the efficiency of the beamlines and ultimately the number of samples that can be analyzed.

ACKNOWLEDGEMENT

The Canadian Light Source is funded by the University of Saskatchewan, the Canadian Institutes of Health Research (CIHR), the National Research Council (NRC), the Natural Sciences and Engineering Research Council (NSERC), the government of Saskatchewan, and Western Economic Diversification Canada

REFERENCES

[1] S. Arzt, *et al*. "Automation of macromolecular crystallography beamlines," Progress in Biophysics and Molecular Biology 89 (2005) p124–152.

[2] S.M. Soltis *et al.* "New paradigm for macromolecular crystallography experiments at SSRL: automated crystal screening and remote data collection," Acta Cryst. (2008). D64, p1210-1221.

[3] G. Snell, *et al.* "Automated Sample Mounting and Alignment System for Biological Crystallography at a Synchrotron Source," Structure, 12 (2004), p537–545.

[4] G. van Rossum. http://www.python.org ,

[5] Django Project. http://www.djangoproject.org .

[6] JSON-RPC Working Group. "JSON-RPC 1.1 Specification", http://www.json-rpc.org.

[7] GTK+ Toolkit, http://www.gtk.org.

[8] The Twisted Framework, http://twistedmatrix.com.

[9] S.B. Pothineni, T. Strutz and V.S. Lamzin. "Automated detection and centring of cryo-colled protein crystals," Acta Cryst. D62 (2006), p1358-1368.

[10] Collaborative Computational Project, Number 4. "The CCP4 Suite: Programs for Protein Crystallography," Acta Cryst. D50 (1994), p760-763.

[11] Collaborative Computational Project, Number 4. "The CCP4 Suite: Programs for Protein Crystallography," Acta Cryst. D50 (1994), p760-763.

[12] G.P. Bourenkov and A.N. Popov "A quantitative approach to data-collection strategies" Acta Cryst. D62 (2006), p58-64.

MECHANICAL VIBRATION MEASUREMENT SYSTEM AT THE CANADIAN LIGHT SOURCE

J.W. Li, E. Matias, Canadian Light Source, Saskatoon, SK, Canada
X.B. Chen, W.J. Zhang, University of Saskatchewan, Saskatoon, Canada.

Abstract

In recent decades, synchrotron radiation has developed into a valuable scientific tool around the world. At synchrotron radiation facilities, the mechanical vibrations in the optics hutch and experimental hutch, especially in the vertical direction, enlarges the beam size and changes intensity of the monochromatic X-ray beam. To investigate mechanical vibrations at the Canadian Light Source (CLS), a vibration measurement system was developed. This paper presents our investigations on mechanical vibrations at four beamlines and endstations at the CLS.

INTRODUCTION

At synchrotron radiation facilities, the vibration of the electron and/or photon beam, especially in the vertical direction, enlarges the size and changes its intensity. This degrades the performance of the beamline. It is reported that the amplitude of floor vibrations at the ATF2 project is approximately 50 μm, which is even larger than the vertical beam spot size expected at ATF2 [1]. In another report related to synchrotron radiation lithography, the quality of micro structures fabricated by the lithography beamline is greatly affected when the amplitude of the vibration is bigger than a quarter of the minimum feature size [2].

Many other factors that are responsible for vibrations at synchrotron radiation facilities were reported in the literature, such as traffic, human activities, strong wind and/or ocean waves, water pipes, and moving mechanical components. Thus, careful investigations of vibrations at synchrotron radiation facilities are crucial, especially if the photon beam size is within a few micrometers.

Studies of vibrations have been conducted at synchrotron radiation facilities worldwide and a brief review can be found in [3]. Although the CLS floor was carefully designed, we found that beamline developments still necessitate carrying out vibration studies. In this study, we investigated vibrations in the experimental and optics hutches at four beamlines and endstations at the CLS: CMCF 08ID-1 beamline, HXMA 06ID-1 beamline, REIXS 10ID-2 beamline, and the STXM endstation at SM 10ID-1 beamline. This work identified key vibration sources.

INSTRUMENTATIONS

The Canadian Light Source Vibration Data Acquisition system includes a Vector Signal Analyzer (VSA) (Model: Hp Agilent 89410A; Manufacturer: HP) and accelerometers (Model: 393B31; Manufacturer: PCB PIEZOTRONICS). Accelerometers produce a voltage proportional to the acceleration of their connected object. The VSA converts the output voltage of the accelerometers into a voltage power spectral density (S_v). Acceleration power spectral density (S_a, unit: $(m/s^2)^2/Hz$) is obtained from S_v by the following equation [4]:

$$S_a = \frac{S_v}{a^2} \tag{1}$$

where a is the sensitivity of the accelerometers. Displacement PSD (S_d, unit: $\mu m^2/Hz$) is calculated using S_a by the following equation [4]:

$$S_d = \frac{S_a \times 10^{12}}{(2\pi f)^4} \tag{2}$$

The RMS displacement over a given frequency band (f_1, f_2) can be calculated using the following equation [4]:

$$Z = \sqrt{\int_{f_1}^{f_2} S_d(f)\,df} \tag{3}$$

The sensitivity of the accelerometer a=1.02 v/(m/s^2). The frequency range of the measurement is 0.1 Hz to 300 Hz. The frequency resolution of the accelerometer is better than 0.1 Hz. In this study, we used two indexes for vibration evaluation--the displacement power spectral density (PSD) and the root mean square (RMS) displacement. The displacement PSD shows the strength of the displacement variation as a function of frequency. The RMS displacement represents the amplitude of displacement variations within a specific frequency range.

IDENTIFICATION OF VIBRATION SOURCES

The experimental set-up is discussed in [3].

Fan coil unit

The fan coil unit is hung on the ceiling in the CMCF 08ID-1 experimental hutch (SOE). Figure 1 shows that the fan coil unit induced vibrations have frequencies of 25.5 Hz (RMS displacement: 2.0×10^{-4} μm), 26.5 Hz (RMS displacement: 3.1×10^{-4} μm), and 53 Hz (RMS displacement: 4.0×10^{-5} μm) which is the harmonics of 26.5 Hz.

Detector cooling system

The equipment is used for cooling the detector of the MicroProbe endstation and it is approximately 0.5 m away from the microprobe endstation in the HXMA 06ID-1 experimental hutch. The microprobe endstation is very sensitive to vibrations since a very small beam spot

(3 μm × 5 μm) is required. Thus, three dimensional vibrations of the microprobe endstation were investigated--particularly, the effects of the detector cooling system on the endstation were studied. In this paper, however, only vibrations in vertical direction are presented. Figure 2 shows the displacement PSD of the microprobe endstation in the z-direction (vertical). RMS displacements in three dimensions are calculated. We found that when the detector cooling system is turned off, the total RMS displacements are 0.001 μm in the *x*-direction, 0.0013 μm in the *y*-direction, and 0.0032 μm in the *z*-direction. When the detector's cooling system is turned on, the total RMS displacements increase to 0.0130 μm in the *x*-direction, 0.0054 μm in the *y*-direction, and 0.0109 μm in the *z*-direction. This suggests that the operation of the detector's cooling system will significantly increase the vibration of the microprobe endstation by 1200% in the *x*-direction, more than 300% in *y*-direction and approximately 240% in the *z*-direction.

Vacuum pump

The Varian TriScroll pumps are widely used as rough vacuum pumps on many beamlines at the CLS. Figure 3 shows the displacement PSD of the floor vibrations in the CMCF 08ID-1 SOE experimental hutch when the Varian TriScroll pump is turned on (red line) and off (blue line), respectively. Figure 3 shows that the TriScroll pump induced vibrations with a frequency of 29.7 Hz (RMS displacement: 1.5×10^{-3} μm) and with harmonics of 59.4 Hz (RMS displacement: 1.2×10^{-4} μm) and 89.1 Hz (RMS displacement: 4.7×10^{-5} μm). Figure 3 shows that the Varian TriScroll pump produces the most significant vibrations (in terms of RMS displacement) on the CMCF 08ID-1 beamline.

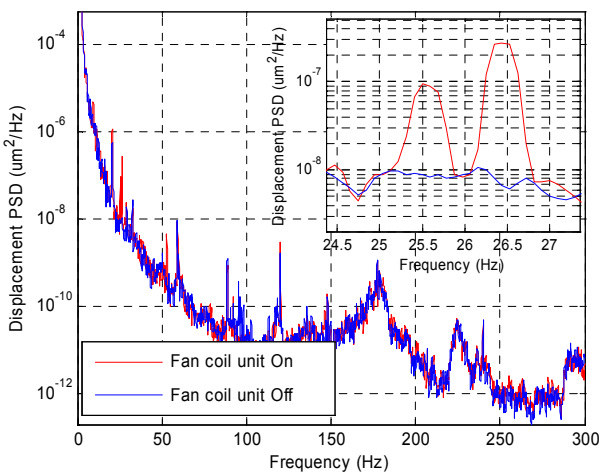


Figure 1: Floor vibrations when the fan coil unit is turned on/off.

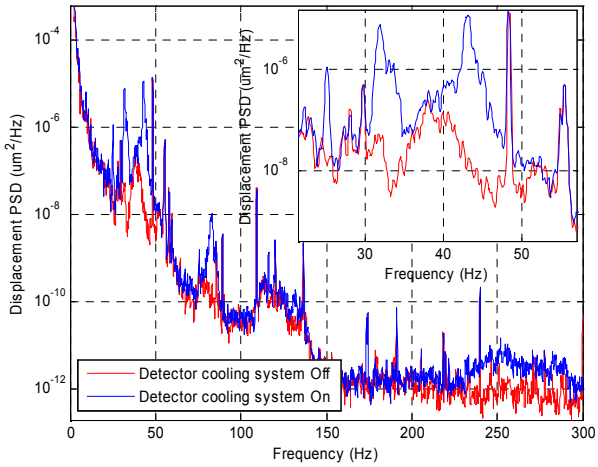


Figure 2: *z*-direction vibrations.

Chiller

The chiller is used for water cooling system and it is approximately 3 meters away from the monochromator outside the optics hutch at the HXMA 06ID-1 beamline. Figure 4 shows that when the chiller is turned on and the damping material is removed, both floor and monochromator vibrations at a frequency of 27.2 Hz dramatically increase, compared to the vibrations when the chiller is turned off. This means when the chiller is in normal operation it causes vibration with frequency of 27.2 Hz and the vibration propagates from the floor to the monochromator. Figure 4 also shows that when the damping material is used, the vibration of 27.2 Hz disappeared from the monochromator and floor. This implies that the used damping material can effectively isolate the chiller induced vibration.

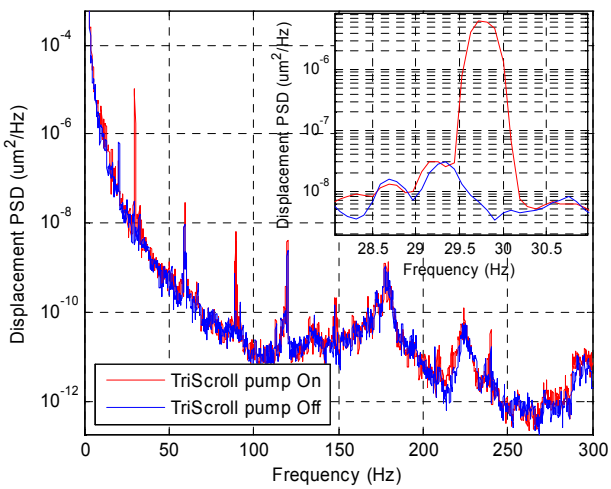


Figure 3: Floor vibrations when the Varian TriScroll pump in SOE is turned on/off.

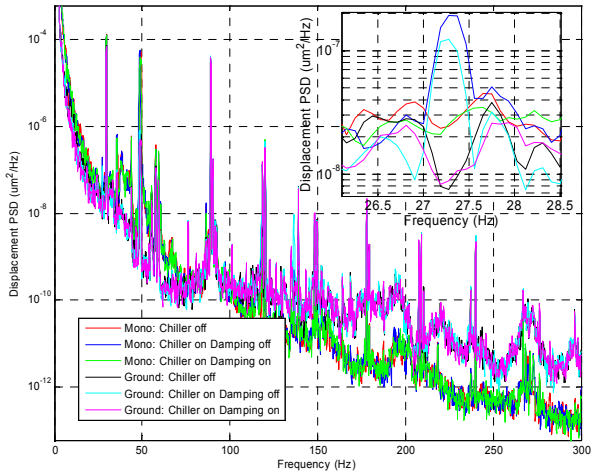


Figure 4: Vibration identification and isolation.

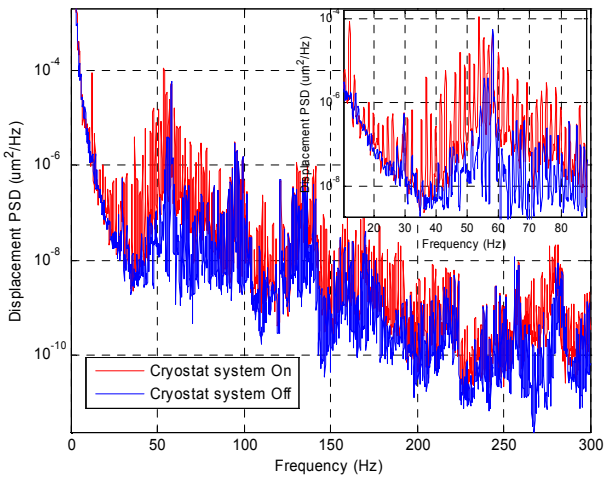


Figure 5: Cryostat system induced vibrations (in vertical direction).

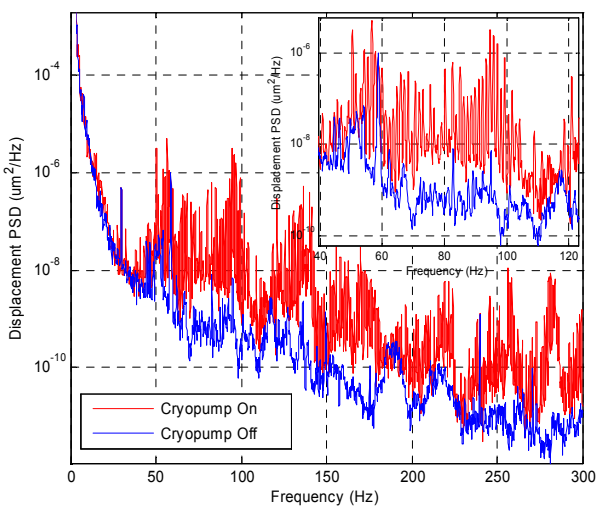


Figure 6: Cryopump induced vibration (in the vertical direction).

Cryostat system

The cryostat compressor is placed outside of the REIXS 10ID-2 experimental hutch and is approximately 1.5 meters away from the endstation The cryostat system includes the cryostat compressor and a cold head inside endstation chamber. The cold head and the cryostat compressor always work simultaneously and thus they are considered as one unit called cryostat system here. Figure 5 shows that the cryostat system produces vibrations with very broad frequency range from approximately 20 Hz to 80 Hz and many of these vibrations have fairly large displacement PSD (over 10^{-6} $\mu m^2/Hz$). Similar observations were found in the horizontal direction, which are not shown in this paper. The cryostat system does not affect the REIXS 10ID-2 beamline so far due to the relatively large beam spot (200μm × 200μm). However, it has been found that its operation significantly affects its neighbour SM 10ID-1 STXM endstation, which is discussed in [3].

Cryopump

The Helix Cryo Torr 8F cryopump compressor is located outside of the REIXS 10ID-2 experimental hutch and just beside the cryostat compressor. Figure 6 shows that the cryopump produces vibrations with very broad frequency range from approximately 40 Hz to 120 Hz, but most of these vibrations have relatively small displacement PSD (below 10^{-6} $\mu m^2/Hz$). Similar observations can be found from vibrations in the horizontal direction, which is not shown in this paper. So far no evidence has been found that the cryopump induced vibrations cause problems for operations of either the REIXS 10ID-2 beamline or the STXM endstation.

CONCLUSIONS

The results demonstrate that mechanical movable equipment in optics hutch and experimental hutch can cause significant vibrations. The information provided in this paper is important to understand and control vibrations not only for beamlines at the CLS but also for other synchrotron radiation facilities worldwide.

REFERENCES

[1] M. Masuzawa, R. Sugahara and H. Yamaoka, “Floor Tilt and Vibration Measurements for the ATF2. IWAA, California, September 2006.

[2] M. Fukuda, N. Endo, H. Tsuyuzaki, M. Suzuki and K. Deguchi, Jpn. J. Appl. Physics. 35 (1996), 6458-6462.

[3] J.W. Li, E. Matias, N. Chen, C.-Y. Kim, J. Wang, J. Gorin, F. He, P. Thorpe, Y. Lu, W.F. Chen, P. Grochulski, X.B. Chen and W.J. Zhang, Journal of Synchrotron Radiation, (submitted in 2010).

[4] J. Paulsen, Vibration Data Acquisition System User’s Manual. Canadian Light Source Inc. Document Number 8.9.44.1. Rev. 0., Approval date: April 28, (2006).

REMOTE ACCESS TO A SCANNING ELECTRON MICROSCOPE USING SCIENCE STUDIO*

D. Maxwell#, E. Matias, CLS, Saskatoon, Canada
M. Bauer, M. Fuller, S. McIntryre, T. Simpson, UWO, London, Canada

Abstract

Science Studio is a web portal, and framework, that provides scientists with a platform to collaborate in distributed teams on research projects, and to remotely access the resources of research facilities located across Canada. The Western Nanofabrication Facility is located at the University of Western Ontario and houses a variety of instruments for lithography, deposition and characterization. One of these instruments is an Oxford Instruments X-ray System fitted to a Scanning Electron Microscope. This x-ray system has been integrated into Science Studio. This allows users to remotely access the system and to upload experimental data into Science Studio. Remote control of the instrument is provided using a remote desktop, so users have access to the full capabilities of the instrument. Through Science Studio, access control and session management are also provided for this instrument.

SCIENCE STUDIO

The Science Studio web portal is an extensible platform that allows scientists to collaborate on research projects, and provides remote access to scientific resources. One resource that is integrated into this system is the VESPERS beamline located at the Canadian Light Source synchrotron [1]. Science Studio provides beamline users with remote access to this powerful scientific tool, and allows experimental data to be easily shared among the project team.

Science Studio is also a framework that can be used to more easily enable remote access to other devices. This framework provides session and experiment management features. Session management allows for remote access to be allocated or scheduled for a specific project team. Experiment management allows the project team to organize and share experimental data. Within the framework is a customizable web portal that provides users a single consistent entry-point for remote access and other services. This web application allows users to manage experiment information and experimental data using in a rich web interface. Security features, such as single sign-on and access control, are also included in the Science Studio framework.

X-RAY MICROANALYSIS SYSTEM

The Western Nanofabrication Facility (WNF) is an open user facility at the University of Western Ontario (UWO) for the fabrication of micro- and nano-structures. This facility has an assortment of equipment and instrumentation that provides its users with a wide range of capabilities; including lithography, deposition, etching and characterization [2]. An instrument of particular interest to users is the LEO (Zeiss) 1540XB Scanning Electron Microscope (SEM) with an integrated Oxford Instruments X-Ray Microanalysis (XRMA) system.

The SEM is a stand-alone instrument with specialized hardware and software for device control and data acquisition. The SEM control software is used for

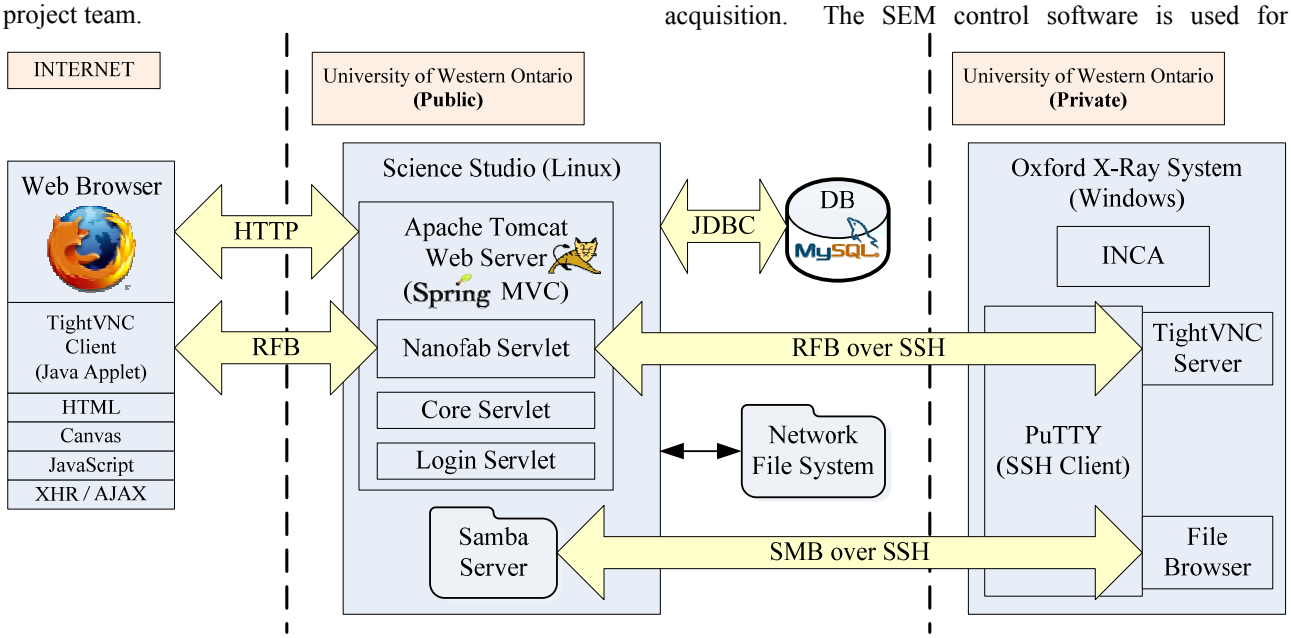


Figure 1: Science Studio architecture for remote access to the x-ray microanalysis system.

*This work was funded by CANAIRE under R&D project NEP-01; and performed at the Canadian Light Source which is supported by NSERC, NRC, CIHR and the University of Saskatchewan.
#dylan.maxwell@lightsource.ca

positioning the sample at the region of interest, focusing at the desired magnification and acquiring high resolution microscopy images.

The XRMA system includes an x-ray detector, which mounts directly onto the SEM, and the required control hardware and software. This system provides elemental mapping and chemical analysis of the sample *in-situ*. The data acquisition and analysis software for the XRMA system is called INCA. The XRMA system is connected to the SEM to enable data acquisition; however, it has very limited control of the SEM. It is this XRMA system that is available for remote access through Science Studio.

REMOTE ACCESS

Remote access to the SEM control software is not required or perhaps desirable as misuse of this software could result in damage to the instrument. An on-site operator must be present to mount the samples into the vacuum chamber of the SEM. This operator will also be responsible for positioning the sample and focusing the image at the desired magnification. In order to allow the remote user to guide this process, using the telephone or another communication method, they are able to observe the SEM control software. This is done using a VGA2USB [3] frame grabber device that intercepts the VGA output of the SEM control computer and converts that signal to a video stream. This device is connected to the XRMA control computer using USB, and the supplied software is used to view the video stream.

INCA is a powerful application for both acquisition and analysis of experimental data. This software is a highly capable spectral analysis tool, with a well designed, user-friendly, graphical interface. Therefore, it is desirable to provide the user will full access to the INCA software.

To meet this objective a Virtual Network Computing (VNC), or remote desktop, solution is implemented to provide the remote user with direct control of the XRMA control computer, and most importantly, the INCA software. The architecture diagram is Figure 1 outlines the components of the remote access system, and their interaction.

Remote Desktop

The TightVNC [4] remote desktop software is used for this project because of its open-source license, excellent performance and availability of a Java Applet TightVNC client. The TightVNC server supports the use of the Mirage [5] video driver, which provides very efficient screen capturing for the Microsoft Windows operating system.

The XRMA control computer is only connected to the UWO private network. In order to allow remote access over the Internet, an SSH tunnel is established between the XRMA computer and the Science Studio server using the PuTTY [6] SSH client.

Special measures are used to ensure the security of the VNC server without further action required by the user. The VNC server port is protected behind a firewall so it cannot accept connections directly from the Internet. The user initiates a VNC session by sending an HTTP request. When the server receives this request, a tunnel is established to forward network traffic between a random port and the VNC server port. The HTTP response contains the random port number, so the VNC client is able to connect to the correct port. This tunnel will only listen for a short period of time (normally ten seconds) for the VNC client to connect, and it will only accept a single connection.

The screen capture in Figure 2 shows XRMA remote access. In the web browser window, on the right, is the remote desktop session. The INCA software is shown with a region of interest defined by a green rectangle on an image of the sample. The video stream from the SEM control computer is visible in the bottom right corner.

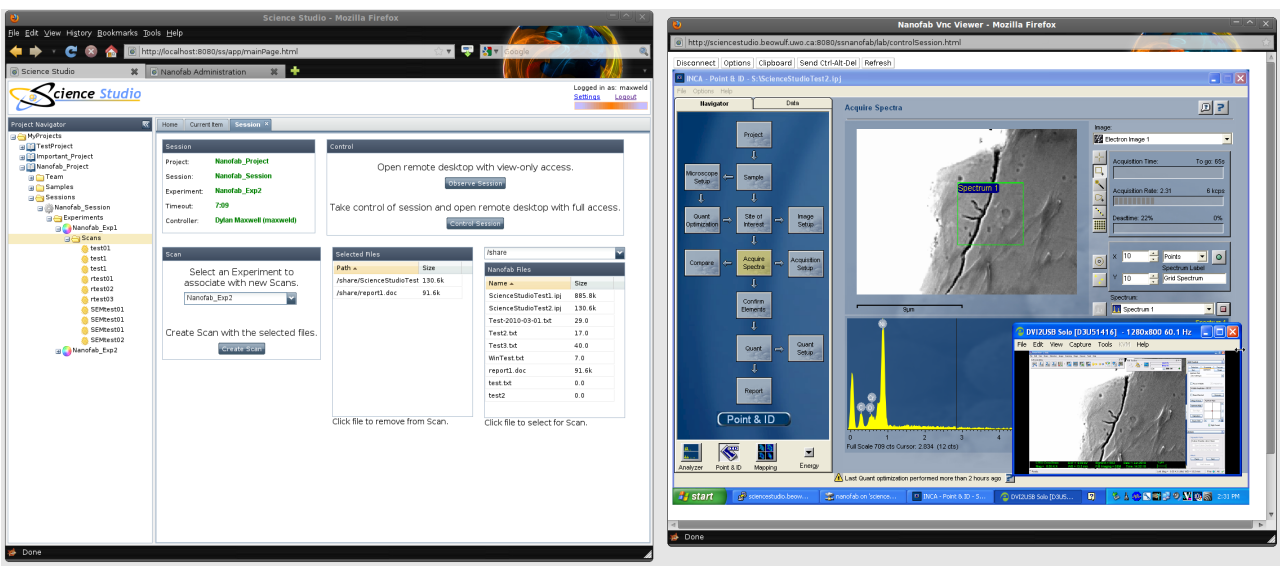


Figure 2: Screen capture of remote access to the x-ray microanalysis system using Science Studio.

Web Application

The XRMA remote interface has two main components. The first component is access to the XRMA computer using remote desktop. The second is a web application that provides the integration with Science Studio. In Figure 1, this is represented by the Nanofab Servlet.

The web application is a rich interface implemented using the Ext [7] JavaScript framework. In Figure 2, the web application is shown on the left. The web application serves two important functions. It allows the user to initiate a remote desktop session with either view-only access or full access to the XRMA computer. And secondly, it allows the user to upload data from the XRMA computer to the Science Studio experiment management system.

When the user acquires data on the XRMA computer using INCA, they save the experimental data files to a network file share that is hosted on the Science Studio server. Then the web application is used to select the files from this share for upload into Science Studio. This will create a scan object within the experiment model. Members of the project team can then access these files by selecting this scan in the project navigator. The project navigator is shown in Figure 2 on the left side of the web application.

REFERENCES

[1] D. Maxwell, *et al.*, "Remote Access to the VESPERS Beamline using Science Studio," PCaPAC, Saskatoon, Oct. 2010, THCOAA02.
[2] Western Nanofabrication Facility <http://www.uwo.ca/fab>
[3] VGA2USB Frame Grabber, Epiphan Systems Inc. <http://www.epiphan.com/products/frame-grabbers>
[4] TightVNC < http://www.tightvnc.com>
[5] Demoforge Mirage Driver for TightVNC <http://www.demoforge.com/dfmirage.htm>
[6] PuTTY Telnet and SSH Client <http://www.chiark.greenend.org.uk/~sgtatham/putty>
[7] Ext JS <http://www.sencha.com/products/js>

CLS USER SERVICES WEB PORTAL*

D. Medrano#, L. Carter, D. Maxwell, CLS, Saskatoon, SK S7N 0X4, Canada

Abstract

The Canadian Light Source (CLS) User Services Web Portal is a collection of web applications that allows users and staff to manage experiment proposals, complete safety training and submit end-of-run surveys. Each user wanting beam time must submit a proposal describing their experiment. Once submitted, the proposal goes through a peer-review process where it is either approved or rejected. All on-site personnel are required to complete safety training. Staff and users are provided with training modules which are completed online. Most training modules consist of two parts: the presentation and the exam. The exams are graded automatically and the results are stored. At the end of each run, users are encouraged to complete an online survey. The survey gives users the opportunity to provide feedback on what was good about their CLS experience and what can be improved to provide them with better service. This paper will give an overview of the design, implementation and capabilities of web portal.

INTRODUCTION

The CLS is an international research facility with a large number and variety of visitors, including students, scientists and contractors. Access to the facility is based on three different roles: users, staff and contractors. Users are scientists that come from all over the world to run scientific experiments on the beamlines, collect data and publish results. Staff operate and maintain the facility and support users in their research. Contractors are personnel hired by the staff for a certain amount of time to complete a task. Contractors can range from labourers to project managers. The goal of the portal is to provide a user friendly and maintainable system to store information associating people with these roles.

PORTAL ARCHITECTURE

Figure 1 shows the architecture of the web portal. It consists of six parts. The first part is the Apache Tomcat servlet container. It is in charge of hosting the four web applications which are the main focus of this paper. These web applications include: training, proposal submission, end-of-run survery and proposal information Application Programming Interface (API). The second part is the Microsoft Active Directory (AD) server. Its purpose is to store all user, staff and contractor login information. Usernames and passwords are authenticated against AD when someone logs into the portal. All communication to AD is done through the Lightweight Directory Access Protocol (LDAP). The third part is the MySQL database server. It contains multiple databases for storing training, proposal, and end-of-run information. Java Database Connectivity (JDBC) is used to communicate with the server. The fourth part is the workflow engine. The workflow engine is used to create the peer-review process a proposal must go through once it is submitted. Workflows are created using the Yet Another Workflow Language (YAWL) [1] and communication is done through the Simple Object Access Protocol (SOAP) [2]. The fifth part is other web systems that make use of the proposal information API. Communication is done through HTTP using Representational State Transfer (REST) [3]. The last part is the web browser which an end-user uses to access the web portal.

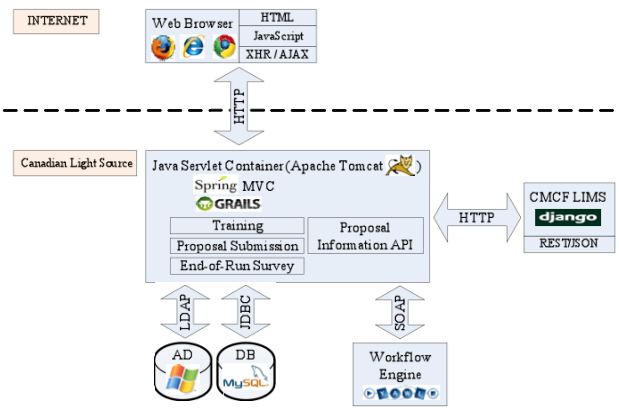


Figure 1: Architecture of the web portal.

TRAINING WEB APPLICATION

Any person coming to or working at the CLS that requires unescorted access must successfully complete and maintain their training. There is a standard set of training modules everyone must complete depending on their role. Users, staff and contractors use the web application to complete training and review training history. When someone takes an exam, a test is generated from a bank of questions in the database. Once a test is submitted, it is marked automatically by the system and the results are recorded. If a person fails the test they must retake it. Staff with the role of administrator use the application to create, edit or delete training modules, generate training reports, create new logins, manually enter training scores and manage training groups and roles.

The training application is written in Java using the Spring Model-View-Controller (MVC) [4] web framework. MVC is an architecture pattern that isolates domain logic from input and presentation. Figure 2 shows the breakdown of the application.

*Work supported by CANARIE under R&D project NEP-01 and performed at the CLS which is supported by NSERC, NRC, CIHR and the University of Saskatchewan.
#Dionisio.Medrano@lightsource.ca

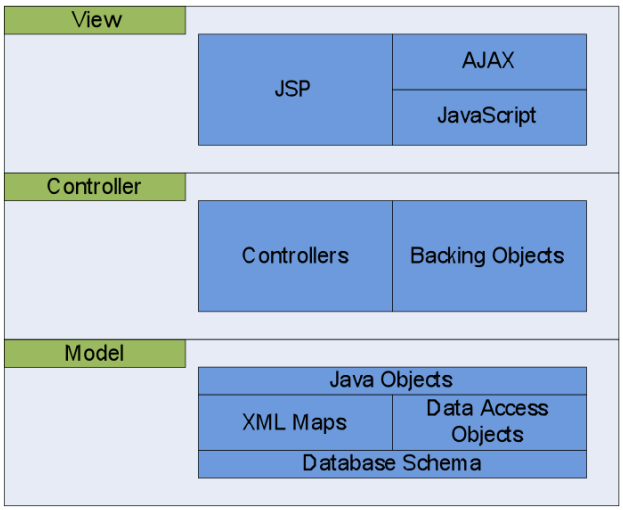


Figure 2: Structure of the training application.

The first layer is the ***Model***. The model is made up of the database schema, XML maps, data access objects, and Java domain objects. The database schema is implemented in MySQL. The domain objects are the Java representation of the database tables and they are mapped together by XML files called maps. This is done with a persistence framework called iBATIS [5]. The data access objects use the XML maps to fetch and store domain objects into the database.

The second layer is the ***Controller*** which contains controllers and backing objects. Controllers are small pieces of code that capture end-user input by handling HTTP requests and performing tasks. The backing objects are Java objects that do not fit in the model but help with getting data to and from the end-user.

The last layer is the ***View*** which consists of JavaServer Pages (JSP), JavaScript and AJAX. The JSPs use the JavaServer Pages Standard Tag Library (JSTL) to render Java objects and data in HTML. The JSPs also use JavaScript and AJAX to provide the end-user with a more rich and dynamic user interface (UI).

PROPOSAL SUBMISSION WEB APPLICATION

The proposal submission application allows users to create and submit proposals. When creating a new proposal, the user must fill out all required information. Examples of the information they must fill out include: type of proposal, name of proposal, funding sources, research team, scientific merit, beamline(s) they wish to use, safety hazards, and equipment they wish to bring. Once the proposal is submitted, reviewers are assigned. The proposal goes through a number of reviews. For example, a beamline scientist does a technical review to see if the experiment is feasible on the beamline. If the proposal passes all its reviews, a final grade is assigned and beam time is allocated. Administrators use the application to manage proposals, beamlines, endstations, equipment and cycles.

The proposal submission application is written in Groovy [6] and Grails [7]. Groovy is a dynamic object-oriented programming language which runs on the Java Virtual Machine (JVM). Grails is an open source web application framework which uses Groovy. The application is divided into three reusable Grails components called plugins. These plugins include clsDomain, clsAdmin and clsStaff. The clsDomain plugin contains the domain model, meaning that it has all the Groovy objects and relationships between them. The clsAdmin plugin contains controllers and views that administrators use. The clsStaff plugin has the controllers and views that staff and users use. Both the clsAdmin and clsStaff plugins use the domain model from the clsDomain plugin.

Grails uses Grails Object Relational Mapping (GORM) to generate the database schema and to map the database tables with the domain objects. This is done by Hibernate [8] which is an object-relational mapping library and persistence framework. Figure 3 shows the database schema generated by the GORM. All the views are done using GroovyServer Pages (GSP). GSPs are very similar to JSPs but use a different tag library.

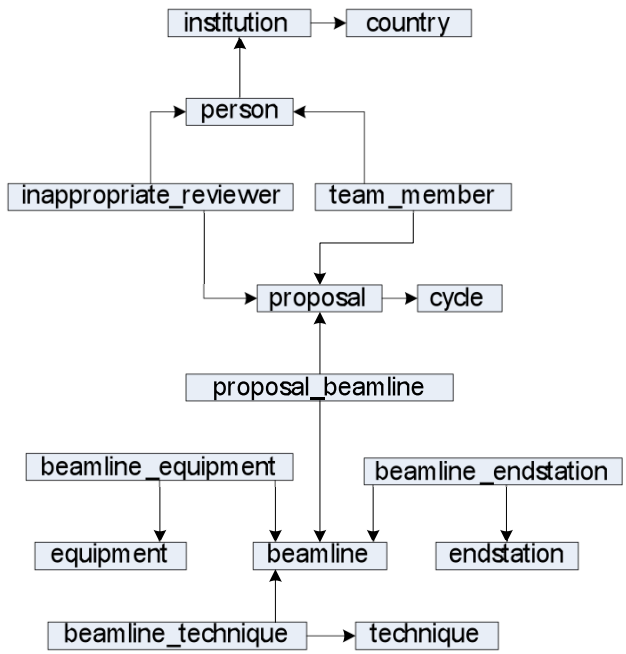


Figure 3: Database schema for the proposal submission application.

END-OF-RUN SURVEY WEB APPLICATION

In order to provide the CLS with feedback, users are encouraged to fill out an online survey about how well their research went while using the facility. The survey consists of a list questions split up into sections. Each question is multiple choice, but there are also some text fields where additional comments can be made. Once a user submits the survey, the feedback is shared with beamline scientists, the Director of Research and

appropriate departments. This helps CLS staff identify areas where improvements can be made to provide users with better service. Administrators gather all the feedback to generate reports for the Users' Advisory Committee.

The end-of-run survey application is implemented using Spring MVC. It has the same structure as the training application. However, the domain model is very simple consisting only of two objects.

PROPOSAL INFORMATION WEB API

There are some other software systems at the CLS that are interested in having proposal information. The proposal submission web application already captures this information so it would be beneficial for it to share it instead of having each system duplicate and store the same information. In order for it to make the information accessible, a web API is being developed.

An example of another system that needs proposal information is the Canadian Macromolecular Crystallography Facility Laboratory Information Management System (CMCF LIMS) [9]. It is a web application for managing sample, shipping and experiment information related to macromolecular crystallography experiments at the CLS. It assists users to prepare crystal samples for shipping, request experiments to be performed automatically, manage experimental parameters, inspect and download experiment results.

When a user wants to do an experiment on the CMCF beamlines, they submit a proposal using the proposal submission web application. Once they have been allocated beam time they use the CMCF LIMS to manage their experiment. The LIMS requires certain information from the proposal, such as, proposal ID and primary contact person information. Instead of having the user enter that information again the LIMS uses the API to fetch it from the proposal submission database.

The proposal information API is implemented in Grails as a RESTful web service. The API uses the domain model from the clsDomain plugin but implements its own controller. When a system wants to fetch data, it sends an HTTP request to a resource backed by the controller. The controller handles HTTP request, looks up the information and formats it in JavaScript Object Notation (JSON) and sends it to the requestor. In order to protect sensitive data, the requestor is authenticated before the request is handled. Currently the API is read-only, meaning that data can only be read from and not written to the database.

SUMMARY

The design, implementation and capabilities of the web portal have been discussed at a very high level. Currently both the training and end-of-run survery web applications are being used in production. The end-of-run survey was deployed in January of 2008 and the training application in January of 2010. Both the proposal submission application and proposal information API are still in development.

REFERENCES

[1] http://www.yawlfoundation.org/
[2] http://www.w3.org/TR/soap/
[3] http://en.wikipedia.org/wiki/Representational_State_Transfer
[4] http://www.springsource.org/
[5] http://ibatis.apache.org/
[6] http://groovy.codehaus.org/
[7] http://www.grails.org/
[8] http://www.hibernate.org/
[9] M. Fodje, *et al*., PCaPAC 2010 Conference Proceedings, THPL005

EPICS DATA ACQUISITION SOFTWARE AT THE CLS

G. Wright, R. Igarashi, Canadian Light Source, Saskatoon, Saskatchewan, Canada

Abstract

The Canadian Light Source (CLS) Data Acquisition library provides a simple scan and store interface for CLS beamlines. Originally intended as a tool for testing and commissioning, it has been used in *QT* and *GTK+* user applications at the beamlines. The current version supports dynamic loading of custom output modules to allow re-definable data transport methods and multiple simultaneous output formats.

INTRODUCTION

During the construction phase of the initial beamlines at the CLS, the need for a simple graphical user interface to perform simple motion control and record results became apparent. An application was written using the GTK+ toolkit. As time progressed, the scanning code was moved into its own library, and this library provides the building block for a number of applications at CLS beamlines. These applications now provide a signification portion of the data acquisition toolkit available at a number of the CLS beamlines.

KEY FEATURES

Configuration

A simple configuration file defines the scan and data collection (**events**).

All data acquisition structures can be accessed and manipulated by the calling program. Standard configuration files can be used to define the initial scan, and with simple data structure manipulation the acquisition can be modified without requiring a custom configuration file.

All Process Variables and range values for acquisition can be specified by macro strings. Again, a standard configuration file can be updated for different data ranges without dealing directly with the configuration.

The data output stream is passed to **Output Handlers** to determine where and how the data is dealt with. This provides the opportunity for customized handling for new data formats or visualization without rebuilding the library.

All scans and events run in independent threads. This allows simultaneous collection of data or recording of large data sets simultaneously with a positioner update.

Scans and Events

The acquisition library has two main components. The first component, the scan, is the definition of the control for the experiment – typically moving a device, and then requesting a detector to detect. A single scan typically only controls a single device. Scans can be nested, allowing multi-dimensional scans.

The second component is the event. An event is triggered from a scan, typically when a detector has finished reading. The event collects data from a list of process variables and requests an Output Handler to deal with the data.

Output Handlers

The section of code that generates the most controversy is the part that defines the output data format. In the data acquisition library, this part of the code has been generalized to a set of function calls that can be set at run time. An object-oriented "factory" for creating links to different handlers interfaces different data formats and different data transports to the acquisition library, allowing a great deal in flexibility on the calling applications' part to deal with new display and storage requirements, and even allow multiple simultaneous data files to be written in different formats.

The second benefit of using output handlers is that any viewer becomes generalized, and can either be integrated directly in the application (as was done for the *Motor Scan* screen in IDA (Interactive Data Acquisition, described later) or streamed to an independent application (such as BLGraph or Grace).

The output handlers have two components: the data format, and the data stream. The data format defines the appearance of the data, whether it's a simple comma separated value text output, or a compact binary output. The data stream is the destination, such as a data file, a named pipe, or a TCP/IP port.

Each instance of an output handler has a set of properties that can be updated through a standard application interface. An application can obtain information on properties and allow these to be controlled directly by the user, so new handlers can be configured without needing to update the user interface software.

Data Visualization

The primary tool used for data visualization with the acquisition library is **BLGraph**. This ROOT-based display tool is highly configurable. Dynamic selection of data fields for display, and manipulation of multiple fields with functions, as well as customizing configurations for quick set up of standard acquisition runs makes this a good match for acquisition library. As well, previously recorded data can be retrieved and viewed.

APPLICATIONS

QT Widget Support

The Qt Widget library from Nokia offers a powerful Rapid Application Development tool (**designer**) and the

ability to create new widgets that inherit from existing widgets to be used directly with **designer**. The acquisition widget has signal and slot support for the main functionality of the acquisition library.

The Data Acquisition GUI

The early separation of the acquisition library from the GUI has, in many ways, allowed more features to be added to the GUI. Initially written as a GTK+ application, the GUI was rewritten using QT.

IDA

The IDA application was written to handle fairly standard synchrotron beamline scans. In addition to setting run parameters for data acquisition, it also includes simple positioner scans (usually used to position a sample before scanning), run time estimates, detector selection, and callouts for energy detuning. The application has no hard-coded Process Variables: it uses a macro definition file to determine Process Variable names. This has allowed quick setup of synchrotron experiments at other beamlines.

IDAV

IDAV is customized for each beamline it runs on (currently SGM and SXRMB beamlines at the CLS). This minimizes the number of applications that need to be running to control and monitor the beamline while data acquisition is in progress.

Science Studio

The Science Studio project being developed at the CLS uses the acquisition library. This is currently in use at the VESPERS beamline.

nD Scanner

The nD scanner grew out of testing the multi-dimension scanning of the acquisition library and the ROOT support for the acquisition library. The application was originally intended as a commissioning tool, but has grown into an application for end-user data collection.

CERN ROOT SUPPORT

ROOT supports dynamic library loading. By providing a C++ class that calls the acquisition library, many of the ROOT features – including a C++ interpreter – are available when using the library. The rich display capabilities of ROOT make for quick work building a custom viewer which extends beyond the capabilities of existing viewers.

RECENT DATA CONFIGURATION FILE CHANGES

One of the most common pieces of code rewritten in each application is an input parser. Using an XML-compliant configuration file gives greater flexibility in input formats, and makes it easier to adapt to future changes in file format. The greatest potential is recording of property values for output handlers, avoiding separate configuration files for the output handlers or having a higher-level application need to rewrite the save-and-restore of output handler properties.

GOING FORWARD

Any system has many opportunities for improvement.

- Despite the gain of a simple configuration file to describe a complex data scan, there are still many opportunities for streamlining the configurations. To this end, support for partial configuration file loading would allow a configuration file per detector (or detector type) that would set up and read back data without knowing the type of experiment, and a partial file that knows how to run a scan for a beamline (but not know the type of detector) to quickly and easily expand the capabilities of a beamline when a new detector is added. The possibility exists that multiple detectors could be easily used together without an additional configuration being manually created.
- Expanding the possible actions to include calls to a language interpreter (e.g. python) would give greater dynamic capabilities to scans.
- Adding a python module to call the library would give benefits very similar to ROOT, but approachable to individuals who are familiar with python but not with ROOT.
- Scan types that aren't based on a start and end value, but rather some other condition would improve the simultaneous scan capability to have a master 'scan' with simultaneous scans that would recognize the completion of the master scan.
- Creating a 'sscan'-type EPICS record would allow embedding the scan in EPICS applications when necessary.

ACKNOWLEDGEMENTS

Jeff Warner provided the initial design work for IDA. David Chevrier created IDAV. The enthusiasm of many beamline scientists provided the inspiration to continuously improve the capabilities of the data acquisition system.

The Canadian Light Source is funded by

- University of Saskatchewan
- Canadian Institutes of Health Research (CIHR)
- National Research Council (NRC)
- Natural Sciences and Engineering Research Council (NSERC)
- Government of Saskatchewan
- Western Economic Diversification

CLS LINAC SAFETY SYSTEM UPGRADE

Hao Zhang, Elder Matias, Grant Cubbon, Carmen Britton, Robby Tanner, Carl Finlay
Canadian Light Source Inc., Saskatoon, Canada

Abstract

The Canadian Light Source (CLS) upgraded the safety system for Linear Accelerator (Linac) in October 2009. IEC 61508 SIL 3 certified components and methods were adopted in the development of the new system. This paper outlines major aspects of the upgrade.

INTRODUCTION

In the CLS, Access Control and Interlock Systems (ACIS) are used in restricted areas to protect personnel from radiation hazards. In the Linac area, a legacy ACIS was used since 1980's until October 2009. The system was based on early Micro84 Programmable Logic Controller (PLC). Given the age of the system, difficulty in procurement of spares as the vendor had discontinued support for the platform; a decision was made to upgrade. Another reason is the old AICS used 120 VAC whereas CLS has adopted 24 VDC for all other control systems. The upgrade ensures the Linac ACIS is consistent with other systems in the facility. All the old sensors, wirings, components, and PLC units were removed. The new ACIS was redesigned and built from scratch.

The new ACIS adopts a two-level, redundant protection mechanism which consists of two independent chains, one governed by a safety-rated PLC system providing SIL-2 as defined by IEC 61508 [1], and a relay-based hardware logic to provide diversity for safety functions.

The system controls access to an area divided into 6 lockup zones [2]. The zone layout was also changed in the upgrade. The zones contain the electron gun, accelerator sections, switchyard, LINAC-to-Booster Transfer Line (LTB), the LTB/Booster Ring (BR1) interface and some adjacent areas including the BR1 RF cavities.

Fundamentally, all lockup zones operate in the same principle, each having its own Emergency Off Stations (EOS), Door Interlock Switches (SWDI), Lockup Stations (LUS), zone lockup lights (ZLL) and horns (HRN).

BACKGROUND

Regulatory Context

CLS holds a Particle Accelerator Operating Licence (PA1OL-02.00/2012) issued by the Canadian Nuclear Safety Commission (CNSC) to operate as a Class 1B facility; as a result the definition of internal process is left to the CLS with the CNSC providing review, oversight, and audition.

Project Plan/Management

The upgrade was carefully planned and documented. The plan identifies project objectives and goals, specifies the upgrade scope, lists standards and guidelines for the development, and defines roles and responsibilities of team members. The plan also includes work structure breakdown, budget, timelines, and a list of documents need to be generated or modified. The plan served as the guiding document during the development process.

Safety System Development Process

The upgrade followed a V-model variant for safety system development.

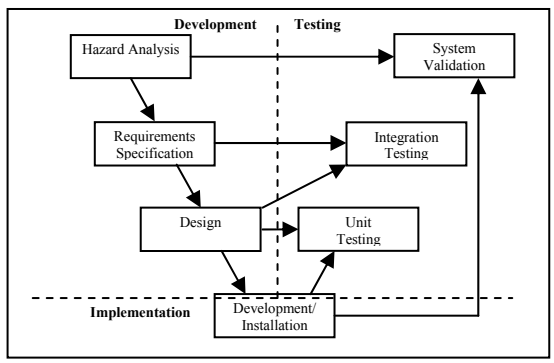


Figure 1: Safety System Development Process

The process starts with the hazard analysis, based on which requirements and specifications are generated, and design and implementation naturally followed from there. Testing was performed in all stages. Respectively, integration and unit testing verify the design meets the requirements and the installation is done as the design.

Hazard Analysis

Since the layout of Linac lockup zones was to be changed, a Hazard Analysis (HAZAN) [3] was necessary to identify the hazards and associated mitigations required with regard to the proposed redesign and upgrade. This was performed by the Health, Safety, and Environment (HSE) department of CLS. The document issued was used as input to the following development stage.

Requirements

The hazards which have been identified and allocated to the ACIS for mitigation in the HAZAN were then examined and refined to generate requirements for the ACIS. Other internal or external guidelines, such as human factor guideline [4] and Canadian Electrical Code were also incorporated as requirements in this stage. Operation experience on the old Linac ACIS and other ACISs was also taken into consideration. A design manual was generated to document all requirements. Linac lockup zone layout drawings were generated to

* Research described in this paper was performed at the Canadian Light Source, which is supported by the NSERC, NRC, CIHR, the Province of Saskatchewan, Western Economic Diversification Canada, and the University of Saskatchewan.

capture detailed requirement and design information. The drawings show zone configurations and lockup paths. All components were identifies and numbered, which makes an IO count possible and perfect input document for wiring diagrams.

FUNCTIONS

The Linac ACIS provides four major functions: *secure*, *lockup*, *annunciation*, and *interlocking*. As mentioned, the system consists of two separate chains, each having their own inputs and outputs.The PLC chain provides all four functions; the relay chain provides redundant functions in safety critical aspects of secure and interlocking.

Secure

A lockup zone is secured only when all the doors are closed and none of the EOSs is pressed. The secure function is implemented independently in both chains.

Limit switches are used to monitor door position. Each door has two physically independent switches for signalling the two separate chains.

An EOS consists of an emergency off button, a reset button, and three mechanically-interlocked and latching contacts - two normally close contacts for signaling the two chains and one normally open contact for activating a local red LED when the EOS is pressed. If the emergency off button is pressed, all contacts remain latched and the red LED remains on until the reset button is pressed.

Linac is interlocked if any of the zones are not secured. The redundant design ensures even component in one chain fails, the other still functions to interlock the Linac.

Lockup

A zone is considered locked up only when the lockup sequence, designed by the HSE for each individual zone, has been performed successfully in this particular zone. Two inspectors are required to perform the sequence, which involves walking through a prescribed path within certain time limit to ensure every part of the zone is inspected in a timely manner.

LUSs are installed in selected locations to ensure the path is followed and the process is timed. Each LUS has a lockup button for signalling the PLC chain, and a green LED to provide visual indication to the inspectors.

As an administrative procedure, the lockup sequence is performed by inspectors and redundantly verified by the PLC. As the complexity of a system increases, so does the potential to introduce errors and possibly hazards. Implementing the multiple sequences in hardware is more likely to introduce error and potential hazards than it is to provide extra protection. Therefore, lockup function is implemented only in the PLC chain.

Annunciation

Horns and flashing lights are used to provide audible and visual annunciations.

Interlocking

Linac is interlocked from both chains through multiple permissive channels, such as Linac RF and gun triggers, RF source switch, etc., to avoid single failure point.

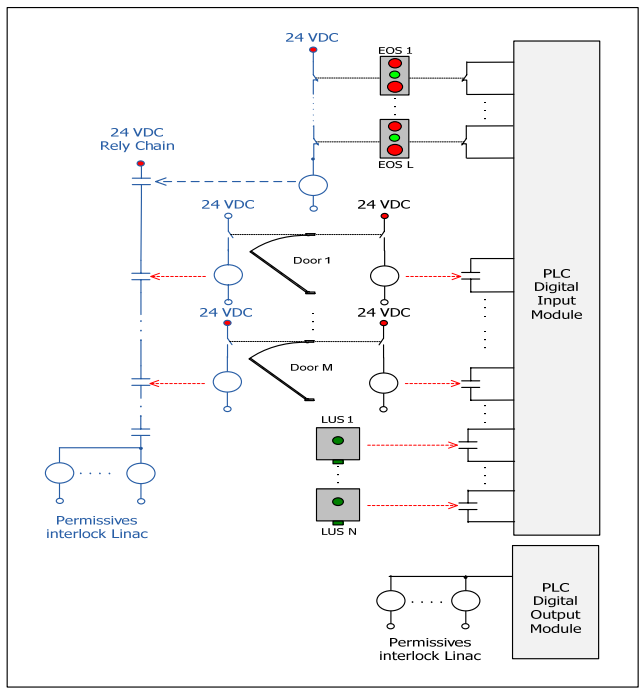


Figure2: implementation of secure and lockup functions

HARDWARE

PLC Configuration

Siemens AS414-4H processor was selected for the CPU. With the fault-tolerant run-time license installed on the processor, the built-in fail-safe run-time logic is activated. Password protection is also activated to protect the processor from re-programming.

SIL-3 certified modules with internal diagnostics and redundant circuitry are used for field I/O. These modules are installed in remote I/O stations communicating with the CPU over Profibus using the PROFISAFE protocol. Fibre-optic cable is used for data link. This configuration is based on accepted practice for SIL-3 applications as per IEC 61508. The protocol is deterministic and failsafe when used with failsafe hardware. The use of distributed I/O via fibre-optic cable provides electrical decoupling of the system, thus avoiding problems associated with running signals over long distances. Given potential problems with ground loops, EMI noise and signal degradation using conventional means, this architecture is more reliable and safe.

Field Wiring

Most of the field wirings are located in the basement Linac hall, where leaking underground water at certain locations can cause problem. Proper NEMA type enclosures were carefully chosen for PLC panels, junction boxes, EOSs, and LUSs to achieve water protection. For

the same reason, field instrumentations are wired using water-proof multi-conductor armoured instrumentation cables, which run in dedicated conduit with distinct color and not shared with other systems or equipment. All field components are CSA approved.

SOFTWARE

The PLC programming toolset is Siemens SIMATIC Manager, using Continuous Function Chart (CFC), a graphical language involving interconnecting elementary Function Blocks (FB) to implement control logics.

Program Structure

The code is structured hierarchically following the actual lockup zone layout. A folder is assigned to each zone, and each zone folder has three CFC charts, which can be considered as programming logical sections. The three charts contain codes to monitor and control EOSs, doors, and lockup sequence respectively. Another folder assigned to control room and contains charters for zone status summation and interlocking.

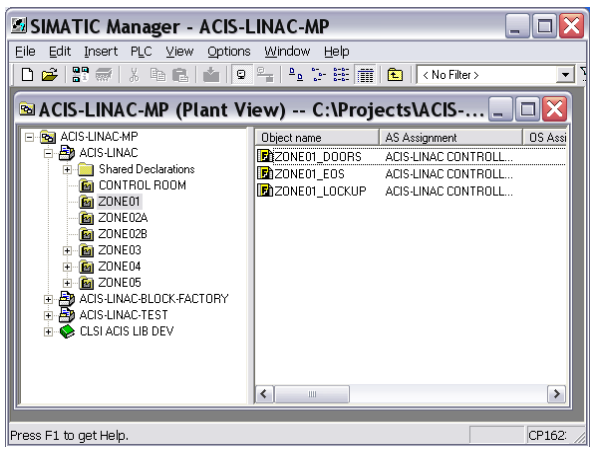


Figure3: Program hierarchy structure

Failsafe Code

Safety critical codes are developed using TÜV-certified function blocks from S7 Fail-Safe Systems Library to ensure fail-safe feature. All failsafe codes are assigned to Organizational Block (OB) 35 by default and are executed cyclically every 100ms in runtime.

Siemens allows developers to create their own standard or failsafe FBs. In CLS, FBs for typical ACIS functions were developed in earlier projects and a CLS ACIS block library are created to save them. In the Linac upgrade, some CLS made FBs were reused and some new FB's were developed and added into the library.

Simulation

The ACIS program had been tested thoroughly using Siemens software simulator, PLCSIM, before it was downloaded to the CPU for on-line testing. Since the system involves only On/Off variables, software simulation is sufficient to test the control logic.

Version Control

For safety system software, it is critical to ensure correct version is loaded on the processor. Siemens S7 F system provides safety program signature to uniquely identify a particular state of the safety program. Generally speaking, a 32-bit number known as the signature is generated across all the fail-safe blocks of the safety program at the end of the compilation phase.

In CLS, MKS Source Integrity is used for software version control. Versions of the ACIS program at different development and maintenance stages are saved in the MKS repository. With the signatures as identifiers, we can easily locate the correct version for download.

VALIDATION AND VERIFICATION

A Validation and Verification (V&V) procedure was developed to examine if the operation of the ACIS within specifications as outlines in requirements and design documents. The overriding approach to the testing methodology is a meticulous and exhaustive series of tests to ensure that the system operates as required. The V&V was performed by HSE personnel before the system was approved for operation. Any modifications to the system after the V&V will cause the V&V procedure being updated and the V&V has to be performed again.

CONCLUSION

The new ACIS was approved for operation in October 2009. Couple of lessons have been learned. Thorough planning and complete documentation in the initial project stage was the key for timely completion. A great portion of early development time went on clearly identifying, defining, and visualizing requirement details in the zone layout drawings and in design manual. This turned out increased the efficiency in the following phases. In the implementation stage, hierarchical program structure provided better readability and made it easier for future Siemens WinCC Graphical User development. Reuse of ACIS FBs reduced programming time. The development of new FBs expanded the block library for future projects. The CLS will continue to use a relay-bsed chain to backup simple, life-safety functions.

REFERENCES

[1] "Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-Related Systems" IEC 61508 or ANSI/ISA S84.01-1996

[2] "LINAC, LTB1, BR1, & HR1 ACCESSCONTROL AND EMERGENCY OFF SYSTEM LAYOUT", CDAC/RAD/0039405, Rev 7.

[3] "LINAC ACIS Upgrade Hazard and Risk Analysis", 11.18.52.1, Rev. A

[4] McKibben, M. 2008. "CLS Human Factors Workscope", 0.1.1.1, Rev. 1

FEC IN DETERMINISTIC CONTROL SYSTEMS OVER GIGABIT ETHERERNET

Cesar Prados Boda, Tibor Fleck, GSI, Darmstadt, Germany

Abstract

Forward Error Correction (FEC) is a technique for recovering from bit errors and frame losses in real-time network applications. Classic recovering strategies, like TCP retransmission, are not suitable due to delay, timing and bandwidth constraints. In this paper, we introduce the FEC technique in a novel deterministic fieldbus, White Rabbit [1] (WR). WR is developed over frame-based computer networking technology, Gigabit Ethernet, GbE. WR provides an effective and resilient way to serve as a deterministic data transfer medium and to interconnect large distributed systems, like Control Systems for Particle Accelerators. The reliability of WR falls on the FEC, which provides the means to guarantee that only one control message per year will be lost or irretrievable as a result of the Bit Error Rate of the physical medium (fiber optic or copper). We propose in this paper a FEC base on LDPC [2], and tailored for broadcast communication in switched networks over noisy channels without retransmission.

INTRODUCTION

Control systems have distributed nodes that need to be connected under specific operation constraints: synchronization accuracy, determinism, bandwidth limitation etc... Besides, the medium over which the communication happens, is a noisy channel where the bits of the frame could be erased or modified. Also, the switches used to propagate the information can mislay or dump such information as a result of collisions in the routing process. So as to ensure an adequate performance of a control system, it has to be endowed with a mechanism capable of overcoming the errors in the communication. Such mechanism is called Frame Error Protection (FEP) and among the different alternatives, in this paper the Forward Error Correction will be discussed. We present the groundwork of an underway research to provide high reliability to time-critical control systems based on GbE and switched networks. The paper is organized in three sections. The first section presents the framework where the FEC is being developed, WR Project, and its boundary conditions. The second section presents how these boundary conditions affect the transmission of data over GbE. In the final section, we analyze the whole scenario and present a FEC scheme to ensure the required reliability.

CONTROL SYSTEMS AND WHITE RABBIT PROJECT

WR is a solution to the generic problem of transferring data in a fast, deterministic and safe manner. WR Protocol (WRP) [4] allows the delivery of timing and control data over a Gigabit Ethernet LAN. WR can be seen as an extension of Gigabit Ethernet, which provides synchronous mode, deterministic routing, bi-directional exchange of frames between nodes and precise delay measurement.

The synchronous mode is achieved by using Synchronous Ethernet along with IEEE 1588, PTP protocol. This combination of protocols provide the means to distribute through the physical layer a common clock within the entire network up to e.g. 2000 stations, allowing 1ns synchronization and 20ps jitter. The frame transmission delay between two stations will never exceed the sum of 64 byte clock cycle plus the propagation time in the longest communication path of the network.

To distinguish between WR and other possible Ethernet traffic in the network, two different frames are defined: SP, Standard Priority frame, which is non-deterministic, and HP, High Priority frame, which is deterministic. The latter frame type is specified in the WRP network to transport messages with the highest priority. HP are frames for time-critical control data, as a consequence, they are routed with lowest latency as possible, forcing fragmentation of non-HP traffic if required. These frames have absolute priority over SP frames and non-WR traffic to maintain low and deterministic transmission delay.

Coming along with the protocol, compliant hardware is being developed in order to support the protocol's features. There are three essential devices: White Rabbit Master, which generates the HP frames and is master clock as well, White Rabbit Switch and White Rabbit Receiver. As a consequence of the device's role and application requirements, the number of units needed in a standard network will consist of one WR Master, M WR Receivers, and $N_{WR_{Switch}}$ with P downlink ports each.

WR allows different approaches to organize the topology of the network depending on the specific requirement of the applications. The strategy for data transmission is based on the distribution from the master to all the other nodes of the network, directly or indirectly according to a Star or Tree topology. The HP frames will be broadcasted from the top of the network, where the WR Master dwells, to the bottom of the network reaching all the WR Receivers.

One of the principal features of the protocol is the notion of determinism, used to guarantee the execution of events within a certain period time. On account of the differential

nature of the time, it is possible to create a slice of time in which everything is perceived as deterministic, what it's called in WR jargon, Granularity Window, GW, [3]. Once we define the span of determinism, size of the GW, WRP provides the means to the transportation of the HP frames and the execution of the events in the very same GW.

Table 1: Granularity Window for 100 μs

Granularity Window	$\approx 5\mu s$	Info Frame Preparation
	t_1	Coding
	$\approx 12\mu s$	Transmission
	t_2	Decoding
	$\approx 12\mu s$	Info Frame Interpretation
	$\approx 12\mu s$	Fail-safe time

GbE AND SWITCHING NETWORKS

Gigabit Ethernet [5] uses as a physical medium optical fiber or twisted-pair cable for sending Ethernet frames. Such frames can be altered due to noise, interference, distortion or bit synchronization errors. The Bit Error Rate or Bit Error Ratio (BER) is the number of bit errors divided by the total number of bits transferred. If a bit error in a frame leads to the complete loss of the frame, the Figure 1 illustrates that a frame would be lost in every $8\ 10^4$ frames sent. It can be also deduced from the figure that small frames are less susceptible to interference, as they are statistically more likely to miss noise caused by internal or external sources.

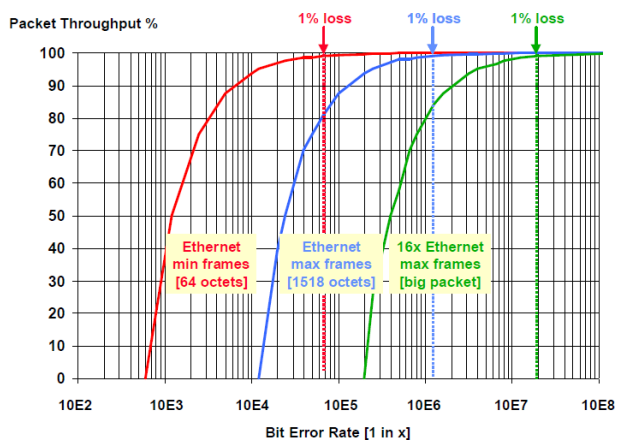


Figure 1: Bit Error Rate in GbE.

The BER can be considered as an estimation of the Bit Error Probability (BEP) in a channel. The sample space Ω of BER will be defined by the collection of all possible outcomes, which means for a single bit:

$$\Omega = E, N_E, Error, Not\ Error \qquad (1)$$

and it is determined by the experimental results of the physical medium.

Broadcast communication is a non thrifty method to convey information in a switched wired distributed system, but terrible effective for simple communication networks. As we presented in the first section, the frames with control events, HP frames, will be broadcasted from the Master throughout the network in order to reach all the WR Receivers, even though the information is not relevant for all receivers. The downside of this approach is a higher global BER. The transport medium that physically consists of a number N of wires, can be considered as an equivalent single cable with a higher BER, as many times as wires are. In other words, the BEP of the system as a whole, is the union of all the probabilities of every single medium path. Since the events defined by the BEP are not mutually exclusive, the union of their probabilities is:

$$BER_{system} = BEP(BEP_1 \cup ... \cup BEP_n) \qquad (2)$$

WR protocol is thought to be a full compliant extension of Ethernet, therefore Cyclic Redundancy Check algorithm is calculated and introduced into the HP frames according to the standard IEEE 802.3 [5]. This field allows early detection of header corruption during HP frame routing. If the header is corrupted, it will be detected and this frame is immediately dropped.

FORWARD ERROR CORRECTION

In the previous sections we presented the scenario for which we are developing a FEP system for data transmission. In short, the master codes the information, adding redundant bits to the frame. This allows the receiver to decode the frame, which implies the detection and correction of errors. In addition the error control has to be able to deal with the following requirements:

- Time constraints due to Granularity Window.
- No feed back channel and not retransmission.
- Stream of HP events within a Granularity Window.
- Recovery of lost and flawed frames.
- Small length of the frame .
- Fully Ethernet compliant.
- One lost frame per year.
- Code Hardware implementation.

The time constrains for WR disqualifies a great number of slow FEC, like Reed Solomon,of which decoding time is proportional to $\theta(k^3)$, with k number of bits in the frame. As can be seen in the Table 1, in a GW of 100 μ, the total time available for coding and decoding is $\approx 70\mu s$. Not only the limitation of time, but also the limitation of upstream traffic, rules out the possibility of positive/negative feedbacks from receivers to sender and the retransmission of a lost o flawed frame, like TCP. This fact reduces drastically the range of suitable strategies. To some extend the GW may limit the length of the frames as well. This reduces the performance of some FEP algorithms that find their optimal

operation with a minimum length. Moreover, the compatibility of WR with Ethernet forces the frame structure, disallowing other suitable organization of the information in the frame. Also, CRC introduces frame losses in the case of an error in the header, disqualifying all the FEP based on one frame transmitted and redundant date on it. The only suitable strategy for WR, capable of overcoming and achieving one lost HP frame per year, is the Forward Error Correction in combination with a repetition strategy.

So as to reckon the magnitude of the problem, we present a case where a WR network is made up of 2000 WR Receivers, WR Switches (1 up-link port and 15 down-link ports each) and one WR Master. There are deployed 144 16-ports WR Switches, 1 up-link port and 15 down-link ports. The connection among WR Switches - WR Switches, and WR Master - WR Switches is established by fiber optic with a BER of 10^{-12}. The connection among WR Switches - WR Receivers is established by fiber optic as well, or copper cable, CAT-5 with a BER of 10^{-10}. The frame consists of 23 bytes in the header and 1000 bytes in the payload. The GW of the system is $100\mu s$, and in every GW only one frame will be sent.

The numbers of cables and global BER of the network is detailed in Table 2.

Table 2: Global BER

	No. FO .	No. CAT-5	BER FO.	BER CAT-5
FO	2144	–	$\simeq 2.144\ 10^{-9}$	–
FO & CAT-5	144	2000	$\simeq 1.44\ 10^{-9}$	$\simeq 2\ 10^{-7}$

Hence, the probability of getting at least one bit error in the header of the frame $P(b_e\_header)$, is expressed by:

$$P(b_e\_header) = \sum_{n\_errors=1}^{bits\_header} \binom{bits\_header}{n\_errors} BER^{n\_errors} \cdot (1-BER)^{frame\_length\ -n\_errors} \quad (3)$$

The probability of getting at least one error in the header of the frame and not in the body can be fairly understood as the Frame Loss Ratio, since a frame with a single error in the header will be always dropped. Through the course of one year, according with the wording of the case, there are $3.145\ 10^{11}$ windows. It leads to assume that within one year the system will suffer $12.4\ 10^4$ losses using fiber optic and $11.5\ 10^6$ using fiber optic and CAT-5.

Table 3: Lost Frames in One Year

	P At least one Error in Header	Frame Lost per Year
Fiber Optic	$3.94\ 10^{-7}$	$12.4\ 10^4$
Fiber Op. & CAT-5	$3.67\ 10^{-5}$	$11.5\ 10^6$

This scenario shows that the coding scheme has to guarantee that a control information frame reaches the receivers even if during the routing the header is been corrupted and dropped. The quick and first answer to this quandary would be to use a repetition scheme. Repetition code repeats bits across a channel to achieve error free communication. Repetition generally offers a poor compromise between data rate and bit error rate. The main attraction of the repetition code is the ease of implementation and straightforward decoding process in case of free errors communication , otherwise, the Maximum Likelihood algorithm has to be used to determine which symbol was transmitted. We have performed simulations where it has been proved that this strategy alone is not suitable. Furthermore, we have evaluated others codes, without success, like Convolutional, LT or Raptor Code. The first code doesn't fullfil our time requirements and the last two codes are protected under patent, or Therefore the current research is aimed to develop a clever scheme of repetition in combination with the codes Low-Density Parity-Check (LDPC) to protect the information as well. LDPC codes is a class of linear block code and are defined by a sparse Parity-Check matrix, H_{mxn}, the encoded bit string, Y_m and a given bit string X_m . This sparse matrix is often randomly generated, subject to the sparsity constraints, which contains only a few 1's in comparison to the amount of 0's.

$$Y_n = H_{m,n} * X_m \quad (4)$$

The main advantage of LDPC is the close performance to the capacity for a lot of different channels and linear time complex algorithms for decoding. Furthermore they are suited for implementations that make heavy use of parallelism. The algorithm used to decode LDPC in our case is the belief propagation algorithm.

The testing implementation of the FEC is developed in VHDL and integrated on the nodes. The test-bed is set up with several WR Switches, providing the networking infrastructure, the WR Master prototype will generate the encoded frames and a WR Receiver prototype will decode and check the integrity of the data. In order to alter the normal behavior of the channel, the cable and nodes will be subjected to artificial noise and errors. From this research is expected to find out and tune up the best parameters for the repetition strategy and the best structure of the Parity-Check Matrix.

REFERENCES

[1] J.Serrano, C.Prados, M.Kreider, R.Baer, T.Fleck "The White Rabbit Project" s ICALEPS'09, Kobe, October 2009, WEP029, `http://www.JACoW.org`

[2] R. G. Gallager. "Low-Density Parity-Check Codes", MIT Press, Cambridge, MA, 1963

[3] Mathias Kreider, Tibor Fleck "FAIR Timing Master", PCaPAC 2010, WEPL011

[4] White Rabbit Switch. Functional Specification. http://www.ohwr.org/projects/white-rabbit

[5] IEEE 802.3-2008 standard

LLRF CONTROL SYSTEM UPGRADE AT FLASH

V. Ayvazyan, K. Czuba, Z. Geng, M. Grecki, O. Hensler, M. Hoffmann, M. Hoffmann, T. Jezynski, W. Koprek, F. Ludwig, K. Rehlich, H. Schlarb, C. Schmidt, S. N. Simrock, H.-C. Weddig
DESY, Hamburg, Germany

Abstract

The Free Electron Laser in Hamburg (FLASH) [1] is a user facility providing high brilliant laser light for experiments. It is also a unique facility for testing the superconducting accelerator technology for the European XFEL and the International Linear Collider. As a test facility, the accelerator undergoes a constant modification and expansion. The last upgrade was started in autumn 2009 and has finished recently [2]. The beam energy is increased to 1.2 GeV by installing a 7th superconducting accelerating module. The new module is a prototype for the European XFEL. In order to increase the free-electron laser (FEL) radiation intensity by linearization of the beam phase space the 3rd harmonic superconducting RF cavities are installed in the injector. The old DSP based LLRF control system [3] has been completely upgraded to latest generation controller boards, down-converters for higher intermediate frequency, algorithms like beam loading compensation, feed-forward waveform generation, etc. are improved. In order to improve the reference frequency signals the master oscillator and frequency distribution system has been upgraded as well.

INTRODUCTION

The FLASH injector consists of a laser-driven photocathode in a 1.5-cell RF cavity operating at 1.3 GHz with a peak accelerating field of 40MV/m on the cathode. The electron injector section is followed by a total of seven TESLA type 12.2 m long accelerating modules each containing eight 9-cell superconducting niobium cavities. The accelerating gradients of the cavities are typically between 20 MV/m and 25 MV/m. Four cavities of sixth module and seventh module are providing gradients above 30 MV/m. The accelerating modules are powered by four RF stations consisting a klystron (tree 5 MW klystrons and one 10 MW multi-beam klystron), a high voltage pulse transformer and a pulsed power supply (modulator). In addition, the RF gun has its own RF station with a 5 MW klystron. The gradient and phase accelerating field (vector sum) of the RF gun and the accelerating modules are controlled by dedicated LLRF regulation system which has been completely upgraded during shutdown period. The FEL radiation is provided by 30 m long undulator section. The undulator consists of periodic structure of permanent magnets which have a fixed gap of 12 mm. The wavelength of the FEL radiation depends on the energy of the accelerated electrons. It can be tuned between 4.3 nm and 120 nm.

After the upgrade a successful operation of FLASH at a wavelength of 4.45 nm has been achieved [2]. For 4.45 nm radiation wavelength the accelerator provides beam energy of 1.207 GeV.

PRINCIPLES FOR LLRF CONTROL

The RF system signal flow is shown in figure 1. The cavity probe signal is converted from the cavity frequency of 1.3 GHz to an intermediate frequency (IF) of 250 kHz for superconducting modules and 54 MHz for 3rd harmonic module. This lower IF holds the original amplitude and phase information of the field inside the cavity.

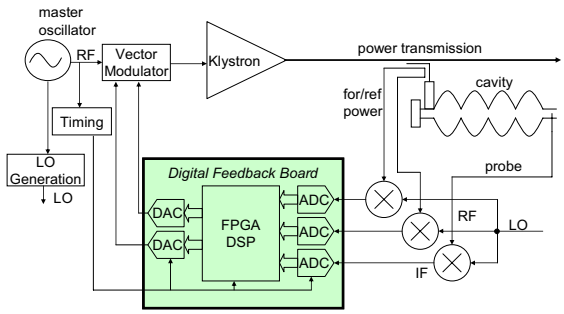


Figure 1: Architecture of the LLRF system.

It is digitized with ADCs (sampling rates of 1 MHz or 81 MHz are used). The digitized signal is going to the digital field detector which extracts the I and Q components out of the input stream. We use two different methods: IQ-sampling and so-called non-IQ-sampling or IF-sampling. The resulting field vector of each cavity is multiplied by a rotation matrix to calibrate amplitude and phases. Finally the field vectors of 8 cavities are summed up for the vector sum of a whole cryogenic module, and those of 2 cryogenic modules are summed up to the vector sum of the RF station which is driven by single klystron. The vector sum of the 16 cavity fields represents the total voltage and phase seen by the beam. This signal is regulated by a feedback control algorithm which calculates corrections to the driving signal of the klystron. The measured vector sum is subtracted from the set-point table and the resulting error signal is amplified and filtered to provide a feedback signal to the vector modulator controlling the incident wave. A feed-forward signal is added to correct the averaged repetitive error components. Beam current information (measured by toroids) is used to scale the feed-forward table to provide fast feed-forward corrections if the beam current varies. The cavity detuning is determined from forward power, reflected power, and probe signal and is used to control the fast piezo tuners to reduce cavity detuning errors to less than a tenth of the cavity bandwidth.

DIGITAL FEEDBACK HARDWARE

After upgrade RF Gun and all accelerating modules are controlled by similar modern FPGA based controller boards with unified firmware and software. The digital feedback hardware consists of Simcon-DSP board (figure 2) which has a VME interface, 10 ADCs to read the intermediate frequency signal from the field probe signals, FPGAs (Xlinnx Virtex II Pro) and DSPs (Tiger Sharc) to execute the control algorithms and 8 DACs, 2 of them drives the vector-modulator for field control. Other components include a timing and synchronization module. The field detection hardware consists of a down converter which converts the cavity field frequency of 1.3 GHz to an intermediate frequency. Additional features included variable input attenuators for level adjustment, an input for a calibration signal and a local oscillator distribution system. The challenging requirements of the down converter are low noise, good linearity over large dynamic range, and small crosstalk.

Figure 2: Simcon DSP board.

DIGITAL FEEDBACK SOFTWARE

The cavity field controller algorithm consists of the field detection scheme (figure 3), calculation of the calibrated vector sum, the field error measurement, the controller filter, a feed-forward signal, and the drive signal generation.

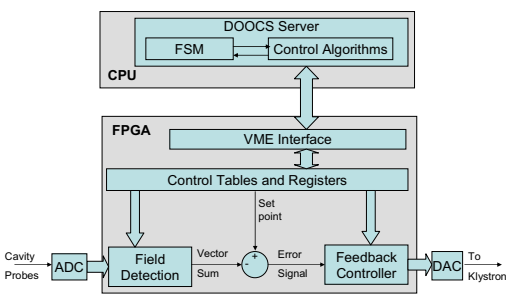


Figure 3: Controller firmware and software architecture.

Beam loading compensation through feed-forward and real time beam measurements are supported. The LLRF control system is integrated with FLASH control system DOOCS [4] by a development of device and middle layer servers. During the shutdown one DOOCS front-end server was developed for all 5 RF stations. Furthermore the DOOCS standard server is used for automation, like simple state machines, and the FLASH data acquisition system for bunch-to-bunch monitoring tasks, e.g. quench-detection.

The control system for the cavities which are driven by a single klystron is considered as a functionally complete unit of the RF system. The feedback algorithm is implemented in the FPGA system. The digital signal processing in turn gets its parameters from the controller server. The controller server software handles: generation of set-point, feed-forward and feedback gain tables from basic settings, rotation matrices for I and Q of each cavity, loop phase constant, start-up configuration files, feedback parameters and exception handler control parameters. The interrupt service routines are used to start the data reading from the controller board. The parameters of the feedback algorithm are modified by the FPGA programs in the time slot between beam macro pulses. It allows a save changing of the parameters of the control algorithm. The functionality of the server gives the user the opportunity to down/upload data into the FPGA (feedback algorithm parameters) and download and start the controller firmware. The server calculates and adjusts the set of the feedback algorithm parameters in accordance with the required field gradient and phase value.

PIEZO CONTROL

The cavities operating with high gradient are deformed due to Lorenz force that causes detuning of the order of the cavity bandwidth from resonance frequency. Detuned cavity reflects the supplied RF power that requires excessive RF driving. For the compensation of Lorenz force detuning (LFD) the piezo actuator is used to excite the cavity mechanically. Each cavity in new accelerating modules (1st, 6th and 7th) is equipped with double piezos that allow compensating of LFD and measurement of cavity vibrations simultaneously.

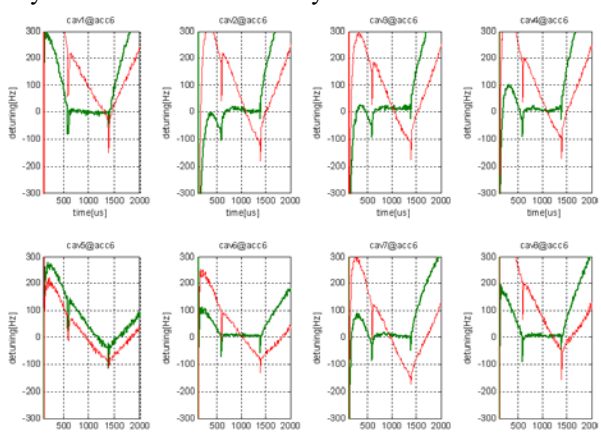


Figure 4: LFD compensation in 6th accelerating module (green – detuning with piezo compensation, red - without)

The piezo control system is able to compute detuning in each cavity basing on RF signals and calculates the

parameters of compensating piezo excitation pulse. The signal from programmable generator is amplified by high power piezo driver. The amplitude of voltage applied to piezo can be up to ±70V and current up to 1A. The results of LFD compensation in 6th module is presented in figure 4. Using piezos the dynamic and static detuning was compensated to only few Hz during flattop in all cavities except the 5th one where piezo is not fixed properly.

APPLICATIONS

A set of generic and especially devoted programs provide the tools for the operators to control the RF system. Some of them are created based on the MATLAB, others, for example, vector sum calibration are implemented as a DOOCS middle layer servers. The adaptive feed-forward is implemented on a front end server, to allow pulse to pulse adaptation.

The application software includes automated operation of the frequency tuners, calibration, phasing of cavities, and adjustment of various control system parameters such as feedback gains, feed-forward tables, and set-point correction during cavity filling. Extensive diagnostics inform the operator about cavity quenches, cavities requiring tuning, and an excessive increase in control power.

Adaptive Control

The RF field regulation is subject to various, random and deterministic disturbance sources. Both disturbance contributions are reduced in closed loop operation by applying a feedback compensator. However repetitive disturbances are particularly suppressed by adaptation of the system input drive, using the known system response from previous pulses. The reference for the RF field is in general not changed very frequently, so the control task can be seen as a repetitive process for the pulsed operation mode of this accelerator. The basic update algorithm [5] is given by

$$u_{k+1}(t) = u_k(t) + L(t)\, e_k(t)$$

where u_k *and* e_k are defined as the system input and the deviation of the measured RF output to the given set-point for the pulse number k, respectively. $L(t)$ is a linear, non-causal, time varying filter based on the identified system model. The current implementation of the system allows changes of all controller tables inside the FPGA between two consecutive pulses. With the minimum computation time necessary for this algorithm, as well as fast data transfer is fast enough, the adaptation can be performed synchronized to the repetition rate of FLASH. Therefore three steps have to be performed between two pulses: Read from previous pulse the error and feed-forward signals e and u, compute the feed-forward signal of next pulse, and write the feed-forward signals to FPGA tables.

MASTER OSCILLATOR AND FREQUENCY DISTRIBUTION

LLRF system provides stable phase reference signals for diagnostics and experiments. The Master Oscillator (MO), which has been upgraded at 2008 already [6], generates various RF frequencies required for accelerator operation. The phase reference system distributes these signals to various locations in the accelerator with low phase noise and very low phase drift. The local oscillator signal is distributed to all of the down converter channels for cavities probe, forward and reflected signals. Typical stability requirements are: 100 fs for short term (few minutes) and 1 ps for long term (several hours).

During this upgrade several MO system components have been improved. The new 1.3 GHz signal generation hardware was installed with improved phase noise and drift performance. The short term stability of about 45 fs was achieved (phase noise integrated from 10 Hz to 1 MHz) directly at the MO output. The temperature coefficient of phase changes demonstrated by the new device does not exceed 200fs/°C, which significantly improved the long term phase reference stability. Additionally, new power amplifier with increased output power and several signal sub-distribution boxes were installed in order to provide the reference signal to bigger number of accelerator devices.

SUMMARY

The FLASH LLRF system regulating amplitude and phase of the accelerating fields has been upgraded to latest generation controller hardware. All modules are controlled by similar modern FPGA based controller boards with unified firmware and software. In addition beam diagnostics signals are in use for fast intra pulse feedback [7]. Algorithms are improved: beam loading compensation, feed-forward waveform generation, etc. For cavity frequency control piezo control has been implemented. In order to improve the reference frequency signals the master oscillator and frequency distribution system has been upgraded as well. FLASH achieved beam energy above 1.2 GeV and lasing below 5 nm with a remarkably improved LLRF control performance.

REFERENCES

[1] W. Ackermann et al., Nature Photonics 1 (2007) 336
[2] S. Schreiber "FLASH Upgrade and First Results", FEL'10, Malmö, Sweden, 2010, 2010
[3] V. Ayvazyan et al., "RF Control System for the DESY FLASH Linear Accelerator", EUROCON'07, Warsaw, Poland, 2007
[4] K. Rehlich "Status of the FLASH Free Electron Laser Control System", ICALEPCS'07, Knoxville, Tennessee, USA, 2007
[5] C. Schmidt et al., "Recent LLRF measurements of the 3rd Harmonic System for FLASH", IPAC'10, Kyoto, Japan, 2010
[6] K. Czuba et al., "The RF Phase Reference Distribution System for The European XFEL", PAC'09, Vancouver, Canada, May 2009
[7] C. Behrens et al., "Intra-train Longitudinal Feedback for Beam Stabilization at FLASH", FEL'10, Malmö, Sweden, 2010

SCRIPTING TOOLS FOR BEAMLINE COMISSIONING AND OPERATION

A. Pazos[#], S. Fiedler, EMBL-Hamburg, Hamburg, Germany
P. Duval, DESY, Hamburg, Germany

Abstract

Scripting tool capabilities are a valuable help for beamline commissioning and for advanced user operation. They are the perfect complement to static Graphical User Interfaces allowing one to create different applications in a rapid way. A light middle-layer for scripting support has been foreseen for the EMBL structural biology beamlines at the PETRA III synchrotron in Hamburg, Germany, to provide 'controlled' rather than 'direct' access to the control system devices. This prevents conflicts with the control system and allows control of the supported operations. In order to account for the wish of different scripting languages by the beamline scientists an extension of the scripting capabilities of the TINE control system has been implemented. To the existing shell support, a Python extension (PyTine) has been added and a Perl wrapping has been also prototyped (tine4perl). An explanation of these implementations and the different wrapping possibilities is also described in this paper.

INTRODUCTION

The EMBL-Hamburg outstation is commissioning three beamlines at the new PETRAIII light source at DESY (Hamburg). In addition, two beamlines at the DORIS storage ring are available for testing and prototyping the arriving instruments.

The control software is based on a client/server architecture integrated with the TINE control system [1]. Each device exports a TINE server that allows its remote operation. Flexibility has been a key feature since the design phase. For this reason different kinds of programming languages like C/C++, Python and Labview™ are supported.

The client side is mainly represented by Graphical User Interfaces (GUI) that connect themselves to the existing device servers. Two kinds of GUIs are available depending of the application. On one side there is an advanced control GUI that allows the operation and tuning of the entire beamline. This is mainly used by the beamline operators and experienced personnel. On the other side there is a GUI for visiting scientists with limited functionality with the main purpose of performing the data collection.

Some procedures, not even supported by the advanced GUI, need to be executed during the commissioning. Moreover, advanced users have the requirement of executing different strategies that are not foreseen at the user GUI.

In both situations the availability of a flexible and rapid way of executing this set of actions is very desirable. For this reason a scripting layer has been introduced at the software architecture allowing one to "glue" calls to the device servers. For gluing and system integration a scripting language can be 5-10 times faster than a system language [2] and the strong typing makes the programs easier to manage.

It is not desirable to the overall operation of a beamline that a user, not familiar with the installed hardware, is allowed to freely execute server functions. Of course, there are control system security measures, but overlaying the servers with a light scripting interface makes the system safer. Thanks to this scripting layer, the naming convention of the functions can be freely chosen.

SCRIPTING REQUIREMENTS

On the basis of our experience with beamline operation and after evaluating the specifications given by the beamline scientists, a list of requirements for the desired scripting environment was compiled:

- Easy to learn (for the developers and for the users)
- Easy to maintain
- Flexible (possible to refactor)
- Dynamic (does not need variable declarations)
- Well defined syntax
- Well documented
- Possible to control the accessible functionality
- Separated from the device specific layer
- Command-line support
- Sequencer support
- Reliable
- Secure
- User proof
- Multi-platform
- Open-source

TINE FOR SCRIPTING

The TINE control system originally supported a set-up of shell commands meant to build shell scripts both in Linux and Windows. Examples for these are the 'tget' (to receive data from a server) and the 'tput' (to send data to a server) commands. These functions are implemented in C and make use of the TINE C API. They receive as an input the necessary information (address, property, data type and data size) to make a call to a server.

[#]apazos@embl-hamburg.de

At first instance, they have been extensively used for commissioning and currently are used for setting up initialization scripts. For experienced users and developers, they allow efficient operation. . However, for users not familiar with the shell environment they might appear cumbersome.

Considering the defined requirement list and adding some extra valuable points (listed bellow), Python [3] was selected as the main supported scripting language.

- It has object oriented possibilities.
- Is getting more popular inside many scientific communities.
- It is also a powerful programming language.
- There are multiple open source libraries available.
- It is also possible to compile and to create executables.
- It is extendable and embeddable.
- There exists graphical support (PyQT [4] and others).
- There is already experience in our group.
- The GUI used at our MX beamline (MxCube [5]) is based on Python.

PYTINE

Initially there was no API for accessing the TINE control system from Python. First ideas where shown at the TINE Workshop, 2007 [6] demonstrating the possibility and the ease of performing such a task. With this starting point an evaluation of the different alternatives was performed.

Native implementation

A native implementation of the TINE control system in Python was evaluated. This would have meant a long term project with complex network implementations. It also would imply a big effort for maintaining and keeping it up to date. This possibility was beyond the scope of the project, having a TINE C API and taking into account that the most-widely used implementation of the Python programming language is written in C.

Python Bindings

The idea was to wrap the TINE C library, implementing Python bindings on top of this (see Fig. 1).

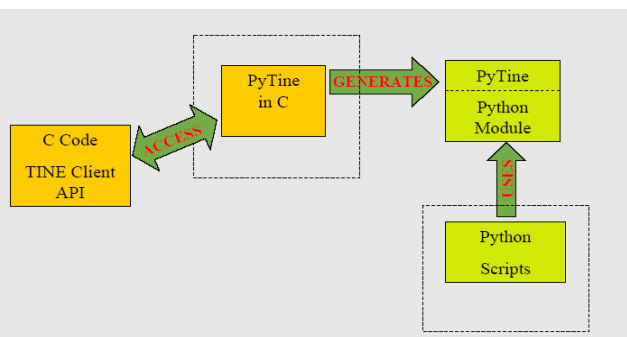


Figure 1 – PyTINE implementation overview

This concept had been successfully used for giving support to other programming languages, such as Labview™ and MatLab™. The use of the TINE Java library was discarded because of better experience of the developers with the C API. The desired outcome was a Python library totally transparent to the C interfaces.

The possibility of using a translator library was also tested. The most popular systems were installed and evaluated: Boost.Python [7] and Swig (Simplified Wrapper and Interface Generator) [8]. The Boost libraries turn out to support more functionality for Python and to be more extended than Swig, but in both cases the translation was not a fully automatic process. For this reason, a native binding inside the C code, without dependencies on a third part library, was decided. This was based on the direct use of the Python.h library and generated with a standard gcc compiler.

All the TINE client functionality was wrapped and a set of new functions was implemented in order to provide a generic friendly interface. This collection constitutes the PyTINE API and its main characteristics are:

- Callback capabilities.
- Support for the TINE data types.
- Data structures available.
- Tested in Linux and Windows.
- Plot functionality integrated, thanks to the use of the PyPlot library [9].
- Integrated inside Labview applications using LabPython [10].

Scripting Middle Layer

As mentioned in the previous section, the PyTINE library is not meant to be invoked directly by the user scripts. It is imported by a set of Python modules, provided by the developers, which create the available functions for implementing scripts. Each of these modules have a specific functionality attached to one or more device servers. They are implemented following an object oriented approach.

In order to perform for example a non-standard data collection, a user can easily implement a Python script. This will call the supported methods of the dataCollection class, which internally take care of the correct operation (see Fig. 2).

```
import dataCollection

//set the exposure parameters
prefix = tst1
dir = /home/marccd/images
run = 1
distance = 320
startphi = 0
phirange = 1.0
exposure = 1.0
frames = 10

# move distance to start synchronous
dataCollection.moveDistance(325)

# start data collection
i = 1
```

```
while i <= frames:
  status = dataCollection.exposeFrame(PHI,
exposure, startphi, phirange, run, dir ,prefix)
 print "Exposing frame ", i , " result: ",
status
 i++

print "Data Collection Finished"
```

Figure 2 – Script to set parameters like rotation angle, starting angle for the phi axis of a diffractometer, detector to crystal distance and initialize a rotating crystal data collection with Xrays recorded by a CCD area detector

TINE4PERL

After the implementation of PyTINE the possibility of interfacing TINE with Perl [11] was also tested. The target was to get and put synchronous data of the basic data types. Making use of the flexibility and extensibility of the control system it is possible to accommodate different developer's flavours regarding programming languages.

Thanks to the experience acquired with the prior implementations, this turned out to be a minor task. Because only the basic functionality was needed and the good support provided for Perl [12], the Swig translation library was selected. To do this, a SWIG interface file (with the extension .i) had to be written. In this file, the ANSI C prototypes that have to be accessed from Perl are listed. In addition, some SWIG directives had to be included. In our case, some specific functions to treat arrays and strings were implemented. Invoking the SWIG command two files are produced: the tine4perl_wrap.c, which contains the C wrapper functions and the tine4perl.pm, which contains the supporting Perl code needed to load and use the module. As last step, the wrapped file has to be compiled and linked into a shared library (see Fig. 3).

```
INCL = /usr/include/tine/
LIBS = /usr/lib

CPP = g++ -fPIC -shared
CC = gcc -g -fPIC -Wall -I${INCL} -c
CCL = cc -g
LM = -lm
LD = ld -G
SWIGPERL = swig -perl5
CCPERL = gcc -I${INCL} -c

tine4perl.so: tine4perl.o
        ${LD} tine4perl.o tine4perl_wrap.o
${LIBS}/libtinemt.so -o tine4perl.so

tine4perl.o: tine4perl.c
        ${SWIGPERL} tine4perl.i
        ${CCPERL} tine4perl.c tine4perl_wrap.c
`perl -    MExtUtils::Embed -e ccopts`
```

Figure 3 – TINE4PERL 'make' commands. It uses a standard gcc compiler and the generated objects to the multithread tine library

CONCLUSION AND OUTLOOK

A scripting language is suited to perform different tasks than a system programming language. We have seen in our applications that if they are used together they can create very powerful programming environments fulfilling complementary requirements.

A scripting language should be as simple as possible. In some occasions it is beneficial not to provide a direct access to the system but to use a middle layer controlling the access to the device servers.

It is important to evaluate very carefully the existing wrapping solutions, including automatic converters, in order to support a new scripting language. Depending on the desired functionality it might be better to use one method or the other. On the one hand, the use of an automatic converter for complex implementations, that possibly include pointers and data structures, it can prove to be a tedious task, making it necessary to learn a special syntax. On the other hand, an automatic converter can create fast bindings for simpler wrappings.

In our environment, where all the software is integrated in a control system, flexible and open systems allow us to extend their functionality and to support new programming languages.

This scripting concept and architecture developed to control synchrotron beamlines could be extended and applied to different instrumental environments and integrated with different control systems.

REFERENCES

[1] P. Bartkiewicz and P. Duval, "TINE as an accelerator control system at DESY", Meas Sci Technol, 18:2379–2386, 2007, p. 2379-2386

[2] J. Ousterhout, "Scriptiong: Higher Level Pogramming for the 21st Century", IEEE Computer magazine, March 1998

[3] Python Programming Language, www.python.org

[4] PyQt White Paper, www.riverbankcomputing.com

[5] J. Gabadinho et al., "MxCuBE: a synchrotron beamline control environment customized for macromolecular crystallography experiments", J. Synchrotron. Rad., 2010, 17, 700-707

[6] D. Franke, "TINE+Python Bindings", (see http://tine.desy.de TINE Workshop 2007).

[7] C++ Boost Libraries, http://www.boost.org/

[8] Simplified Wrapped and Interface Generator (SWIG), http://www.swig.org/

[9] Matplotlib, http://matplotlib.sourceforge.net/

[10] Labpython, Open Source Python tools for Labview™, http://labpython.sourceforge.net/

[11] Perl Programming Language, http://www.perl.org/

[12] D. Beazley et al., "Perl Extension Building with SWIG", O'Reilly Perl Conference 2.0, 1998, San Jose, California

THE ANKA B-FIELD TEST FACILITY CONTROL SYSTEM, BASED ON A SPEC MACRO PACKAGE ENHANCED SETUP*

Karlheinz Cerff, Thomas Spangenberg, Wofgang Mexner, Institut for Synchrotron Radiation, (ISS)-ANKA, Karlsruhe Institut of Technology, (KIT)-Campus North, Germany.

Abstract

The ANKA B-field test facility provides users with a flexible tool to investigate magnetic field distributions of different setups of coils or permanent magnets, optimal sensor types, geometrical alignments of probes and the possibility to change the independent physical stimuli to generate and alter magnetic field distributions [1]. From the point of Software development it is taken as an example of a straight-forward device implementation with a recently introduced type of macro based 'building block system' for devices in SPEC, [2]. This macro package provides the C-like SPEC with an object orientated framework with a namespace and class concept to represent the power supplies of different brands, probe positioning devices and measurement amplifiers.

INTRODUCTION

The B-Field Test facility provides measurement data of magnetic field distributions of coils or permanent magnet structures, within the range of um spatial resolution, over positioning ranges up to meters, devices in use are,

- a stepper motor driven, encoder monitored linear positioning probe, equipped with a variable geometrical arrangement of Hall-sensors to measure B-field induced voltage gradients.
- Two power supplies, consisting of a main and a second, multiple power supply, driving individual shaped I-current ramping functions for corrector coils.
- A Digital Multi-Meter (DMM) of Keithley, type 'k2700' to read out, up to n Hall-probes.

The control software package should also generate a raw data fit for a polynomial of variable degree i (i<=9), for up to n Hall-probes. At last the control system monitors the safe operation of the Test facility, for example it shuts down the main power supply when a superconducting coil under test is quenching.

IMPLEMENTATION

In the context of the 'Macro package based Enhancemend of SPEC controlled Experimental Setup'[3], this means that the device properties are stored as elements of data structures (SPEC global associative arrays). The task of the software development is, to

- set up an abstract model of the B-Test Facility hardware devices.
- write the device drivers for B-Test Facility motor, power supplies and digital multi meters.
- linking the resulting SPEC macro functions to the Interface generated by enhanced macro package.

The introduction of a set of interfacing rules minimizes the risk of damage to existing SPEC-structures, furthermore it opens the possibility to port in this way generated SPEC-'classes' to other experimental facilities.

Table.1: B-Test facility, list of realized implementation of functions, devices, SPEC '-instances' and –'classes'.

physical function	device	SPEC-'instance'	SPEC–'class' (macro)
motor controller, one channel	OMS-Maxv	'm0'	Motor.mac
main power supply, 1 channel.	FUG NTV-1000	'fugbig'	Fug.mac
power-supply small, 8 channels	FUG NTV-100	'fug'	Fug.mac
Digital multi-meter Hall-probes	Keithley, K2700/7703	'k2770' 'Hall n'	Anka-Keithley.mac

Setting up the B-Test Facility, the two power supplies are defined as members of the 'class', represented by FUG.mac. They are both instantiated as objects 'fug' and 'fugbig' in the declared global associative array 'FUG', writing a set of device dependent standard-values to it. SPEC-associative Arrays offer as possible arguments arbitrary strings or numbers instead of integers [2]. In the 'class'-macro keithley_anka.mac, the Keithley DMM is instantiated as object "k2700" and the connected Hall-probes as objects "Hall-1"-"Hall-15. The minisetup class' macro contains the 'standardvalues' declarations and a data fit object to fit raw data to a polynomial up to the order of nine.

Benefit

- Two FUG devices, representing nine power supply 'objects' can be accessed by 11 (for the main power supply) and 73 (for the corrector power supply) standard-function calls obeying the naming rules introduced by the macro package.
- Up to fifteen Hall-probes have to be addressed by 255 standard function calls for the Hall-probes plus three functions for the K2770.

The advantages using the object oriented approach is clearly visible, there is no need to write, a set of 84 nearly identical conventional SPEC-functions for power supplies and additional 255 functions to handle the output, in addition existing ANKA-beamline driver modules for motors can be used.

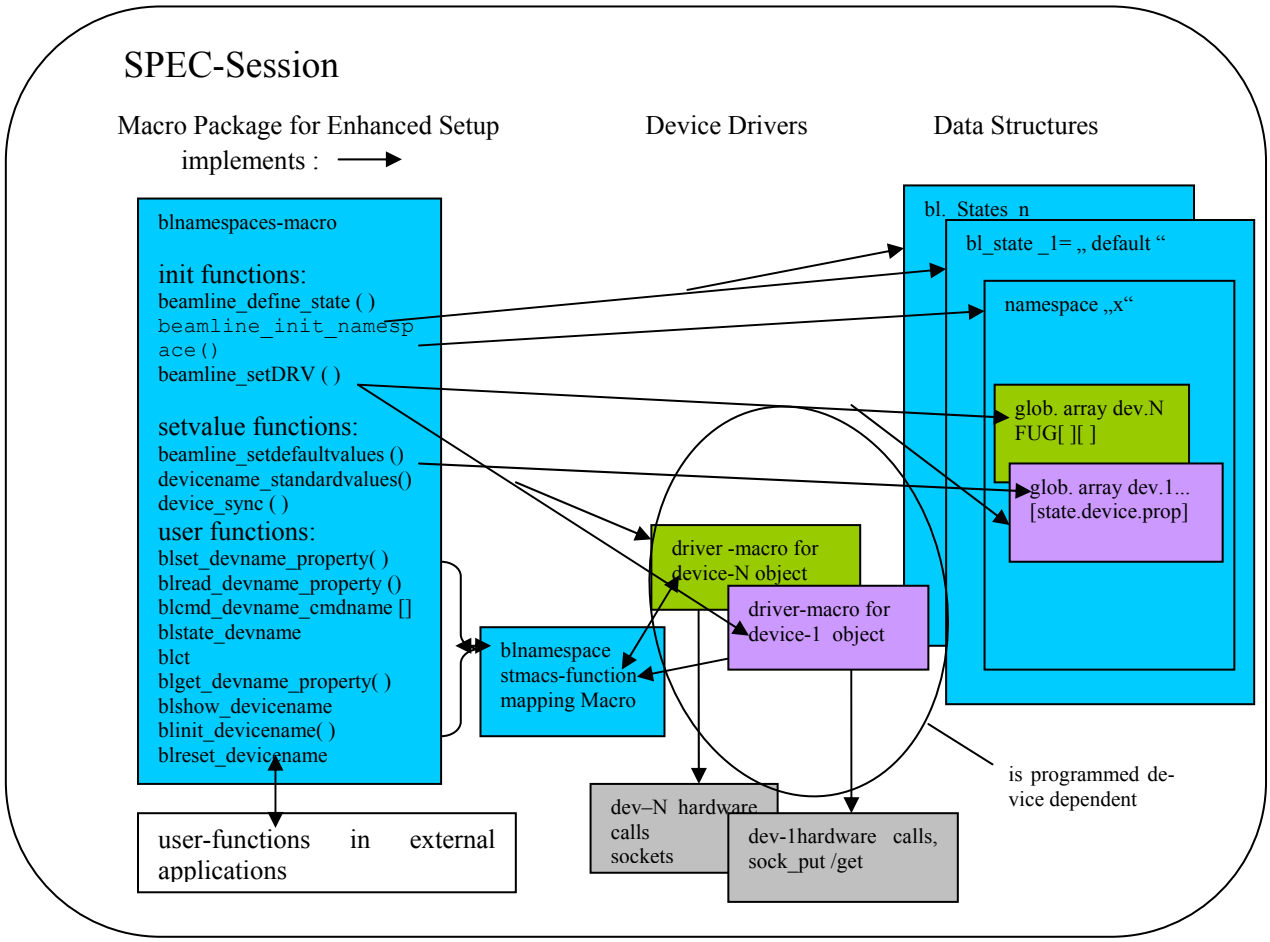


Figure 1: Macro Package generated structures (blue), driver software to be written (green and cyan), SPEC-built in functions (grey), interface function calls (white)

BUILDING THE B-TEST FACILITY DEVICE MODELS

Loading and executing, the blnamespaces-macro, which is the heart of enhanced setup, submits the functionality for setting up the namespace and global array structures:

beamline_define_state (“x”,”default”*)*
beamline_init_namespace (“x ..”*)*

Both function are processed only once, because all devices are instantiated in one state “default” and namespace “x”. In principle the concept allows multiple state definitions “others” which could be used for example to define different arrangements of Hall sensors The functions below instantinate the device-objects given in the first column:

beamline_setDRV(“fugbig . ”,”FUG”)
beamline_setDRV(“fug . ”,”FUG”)
beamline_setDRV(“k2700a . ”,”KEITHLEY”)
beamline_setDRV(“hall n . ”,”KEITHLEY”)

The power supplies are abstracted by:
status, ramping behaviour, address, type, set/get/ voltages, I-currents, I-current-rates. The device models are stored as sets of object variables in the associative arrays “FUG” and “KEITHLEY”, generated by the macro package init functions, s. Fig.1:

devn . property = “value” structure:

```
FUG["fug"]["$active"] = 0
FUG["fug"]["$adress"] = "192.168.4.4:23"
FUG["fug"]["$fugtype"] = "FUG-NTV 100"
FUG["fug"]["$maxcurrent"] = 10
FUG["fug"]["*current1"] = 0
FUG["fug"]["currentrate 1-n"] = 0.2
FUG["fug"]["dcpower 1-n"] = 0
FUG["fug"]["readout 1-n"] = 1
```

The prefix in the second array elements marks the state of properties: “private”, “read only”, “read/write” or “command “.

B-TEST FACILITY DEVICE DRIVERS

The program code which has to be written are the device driver macros for power supplies “fug” and “fugbig” and the digital multi meter with connected Hall- probes.

The functions can be grouped in :

- internal functions, like socket functions to set/get specific hardware register values, to reset or initialize devices, to address sub device and functions to process data strings received.
- Functions for data synchronisation with the pre-defined standard values in associative arrays or with the ongoing values of the hardware device of interest,
- functions to set /get device parameters by calling external measurement devices used at ANKA-beam lines
- functions, which are 'built in' SPEC, here used for the linear motor drive with encoders to position Hall-probes.

FUNCTION-MAPPING

A set of 'standard-' or user functions' for communication is generated automatically by the macro package. The bulky type of driver functions with long argument lists is mapped to a set of user friendlier functions. The functions have the general form:

```
def user_function (value, argument ) '{
<return> driver function ("device name . property")
}
```

The simplest user functions don't have arguments, for example a 'blct'-call, gives the outputs of all parameters of the assembly of power supplies, DMMs, and Hall-probes of the B-Test facility:
A generic example for function mapping, will be the 'setcurrentrate' user function for 8-fold power supply 'fug', device No 3, with a I-current rate of 0.2A/sec.
The user function call is ,

- *blset_fug_currentrate3(0.2)*

mapping to the device driver function :

- *FUG_setcurrentrate3("fug", 0.2, 3).*

This calls the SPEC socket functions of the driver to write an appropriate value to the hardware register sub address 3 of the power supply "fug", after command reference given in [5]:

```
def FUG_setcurrentrate3(device,quiet,value,) '{
__FUG_setcurrentrate(device,quiet,value,2)  }'

   # call of __internal driver function :
def __FUG_setcurrentrate(device,quiet,value,devnr) '{
   #which type of power supply ?:
if ( FUG[device]["$fugtype"]=="FUG-NTV 100") {
   # write external inputs for ps with devnr=2+1 to'
    value':
value =  NumberInput ("current rate", FUG [device]
[sprintf ("setcurrentrate%i",devnr+1)] ,0, 1, quiet,
value);
   # call subdevice 3, addressing, convention, s. command reference [5]
__FUG_sendcommand(device,sprintf ("%s>S%iR
%g\n",sprintf("#%i",int(devnr/2)),devnr-
2*int(devnr/2), value ));
   # The __internal function uses the basic 'built in'
    SPEC socket_put function:
def __FUG_sendcommand (device,command) '{
  sock_put(FUG["device"]["$address"],command);
        }
}'
   # value gets the formatted readback from subdev. 3:
value = __FUG_splitanswer(__FUG_readback
(device));

   # __internal function calls basic  sock_get function:
def __FUG_readback(device)  '{ local tmp;
     tmp=sock_get(FUG["device"]["$adress"]);
       }'
   # updates appropriate element of global array FUG
with current read back value:

FUG[device][sprintf("setcurrentrate%i",devnr+1)]=_
_FUG_readcurrentrate (device, quiet ? 0 :1,devnr);
 }
```

CONCLUSION

The object oriented implementation, by use of existing beam-line software modules make the procedure straightforward since only the missing drivers for power supplies, digital multi-meters and the raw data evaluation algorithm, have to be introduced. But synergy proceeds, the FUGs will be the power supplies of future insertion devices [4] at ANKA, so the Software modules written to control its devices can easily be ported to the control system of the next ANKA superconducting undulator.

REFERENCES

[1] *CASPER*- A magnetic measurement facility for superconducting undulators,
E Mashkina et al *2008 J. Phys*.: Conf. Ser. 97 012020
[2] www.certif.com, software SPEC
[3] Macro Package based Enhancement of SPEC controlled Experimental Setups, T. Spangenberg, K. Cerff, W. Mexner
Proceedings of PCaPAC2010, Canadian Light Source, Saskatoon, Canada , October 2010
[4] A modular control system based on ACS for present and future ANKA insertion devices
K. Cerff, W. Mexner, T. Spangenberg, M. Hagelstein,
Proceedings of PCaPAC2008, Ljubljana, Slovenia, October 2008
[5] FUG, Probus, ADDAT30, command reference, V2,13

MACRO PACKAGE BASED ENHANCEMENT OF SPEC CONTROLLED EXPERIMENTAL SETUPS

Thomas Spangenberg*), Karlheinz Cerff, Wolfgang Mexner
Institut for Synchrotron radiation, ISS, ANKA, KIT-Campus North, Karlsruhe, Germany

Abstract

Certified Scientific Software's program package spec [1] for X-Ray diffraction and data acquisition provides reliable instrument control to scientists at synchrotrons and other facilities worldwide. It's very flexible C-like macro language provides a large number of degrees of freedom for experiment control as advantage and as big disadvantage at the same time. A large number of programmers with their own ideas and naming conventions are contributing to the growth of functionality. At the same time the risk of collateral damage by accidentally overriding already existing functions and variables grows constantly. To solve this dilemma a new object oriented like software development concept for spec is proposed. A few naming rules plus a macro package in combination with a single client-server-application expand the manageability and options to control experiments considerably. As main goal spec gets an object-like handling and a standardized user interface of newly introduced devices. A generic server-client based interface allows a smooth integration of spec in more complex control environments via TANGO [2].

INTRODUCTION

Most of the physical and logical devices provides the opportunity to operate them in a simplified model as a set of independent properties which are offered by a certain remote interface. Therefore it becomes possible to integrate them rapidly into its own measurement setup either by direct driver support or by some macro integration.

As an example, the software package SPEC with its flexible macro language and various interfaces offers a number of paths to implement additional hardware into an experiment.

It will be shown that the risk of interfering solutions can be avoided for the device integration by introducing a few design rules in combination with a macro package. Additionally the client server based export possibilities of the integrated devices will be increased significantly.

MACRO PACKAGE AND DATA STRUCTURING

The basic idea of that macro package is to organize and handle devices object like although SPEC's pure macro based programming language definition doesn't support objects directly. But the provided data structures permit with a few limitations an object like structuring of data and a macro supported creation of specific functions to manipulate them.

Starting from the abovementioned simplified device model the representation of the device properties is stored into SPEC's associative arrays (see Fig. 1) which yields three advantages.

- First, all objects of one class are stored in only one array variable. It is evident, that a naming conflict can be prevented by using a single identifier per class.
- Second, due to SPEC's data type definition any type of data can be stored into this array.
- Third, the two dimensional index organized by strings is well suited to store data differentiated into 'objects', their properties, and their methods.

The data organization of the macro package is basically funded to associated arrays and is introducing a naming convention to their indices. SPEC defines associative arrays as a string indexed data object which stores any type of information. The first dimension of the two dimensional index is used for the device name. The name is usually chosen as an acronym which describes the device function in the experiment (e.g. vc1 for vacuum controller 1, see fig. 1).

The second part of the index string is primary subjected to the device property. Additionally the first character is used to transport the minimal necessary information about the represented property which is used for the automatically generation of the user interface. The implemented scheme is as follows:

- '$' indicates internal variables. There are no user functions provided.
- '*' indicates read only properties or variables. Read functions are provided.
- '!' indicates a command. A command function will be available.
- no special character indicates a read/write property. Read and write functions are provided.

The formal initialization overhead due to the macro package is very small. There are 2 functions for the whole macro package and only 3 additional steps are needed to implement a new device. There are:

- The formal declaration of the device instance by name and device type, followed by
- The initialization and declaration of start-up values and finished by
- Initializing the device or synchronizing the stored information.

The macro package evaluates the stored data and creates automatically the functions to manipulate them obeying the fixed naming scheme. Thereby the whole user functionality will be generated.

*) thomas.spangenberg@kit.edu

EXAMPLE IMPLEMENTATION

An example may demonstrate the situation. Assuming a hypothetical vacuum pump controller (similar as shown in fig. 1) device vc1 which may store its data into the associative array VPC. The declaration of that structure is done by global VPC while loading the macro package for that type of controller.

private: address
r/o: current
r/w: voltage
command: on/off

Figure 1: an example vacuum pump controller and its properties

As it is common to object orientated approaches a set of 'class-functions' needs to be provided by the controllers macro package. The first argument of all functions is the device name. Other arguments are regulated and straight forward connected to the idea of the object like access and the simplified device model approach.

The value initialization is done by VPC_standardvalues(device, [arg1 [,arg2...]]). This special function (see fig. 2) doesn't have strong naming rule because it will never be used automatically by the package and depends of course from the device which is to be implemented.

The task of the function is comparable to a constructor of an object. It has to pre initialize all instance variables and at the same time it is declaring the user interface functionality due to the fixed naming scheme abovementioned.

```
def VPC_standardvalues(device,address) '{
VPC[device]["$adress"] = address
VPC[device]["*current"] = 0
VPC[device]["voltage"] = 0
VPC[device]["!on"] = "VPC_poweron"
VPC[device]["!off"] = "VPC_poweroff" }'
```

Figure 2: example device value initialization

Furthermore current implementations of the macro package expecting the functions VPC_init(device) which drops all pre setted values into the device, VPC_sync(device) to synchronize the object to the device otherwise, and VPC_state(device) which is printing the read device state onto the screen.

Declared commands are realized by any function which has to handle 2 arguments. The first is the device and the second is the optional user argument. An example may be VPC_poweron(device,option), which name was stored into the device property '!on'.

For reading and setting the property XYZ the functions VPC_readXYZ(device,..) and VPC_setXYZ(device,..) need to be defined.

The read and set functions have to provide some other arguments which will be discussed following.

FUNCTION ARGUMENTS

The macro package requires from all 'class-functions' a strict organization of all arguments concerning their order and the meaning. The first argument is always the device name.

Reading and setting functions are already differentiated by the second argument which is for reading functions an integer indicating the verbosity of it. The complete declaration of the example read function is as follows VPC_readXYZ(device,verbose).

The argument verbose regulates the verbosity which can be switched on or off.

Setting functions using as a second argument an integer which lets them operate quiet. In that case the third argument represents the value to be set. The declaration is therefore

VPC_setXYZ(device, quiet, value)

It is quiet clear that these 'driver class functions' needs to be programmed with respect to the device and are therefore similar to other approaches in relation to the necessary programming effort. The goal are the generated user interface and the export capabilities.

USER INTERFACE

Just offering to the user device view orientated functions doesn't satisfy the users view to an experiment, which is usually more orientated to the job that needs to done than to a certain device.

The macro package evaluates automatically the array stored information (the index names and '$*!') and builds the whole set of corresponding functions and macros which are representing the user interface for any device. Even different user custom is satisfied by creating a function based access and a macro based access as well at the same time.

The created set for the example is shown in table 1:

Table 1: corresponding set of device functions and generated user functions

device function	user function / macro
VCP_state("vc1")	blstate_vc1 blstate vc1
VCP_readXYZ("vc1",...)	blread_vc1_XYZ(...) blread vc1.XYZ
VCP_setXYZ("vc1",...)	blset_vc1_XYZ(...) blset vc1.XYZ
VCP_poweron("vc1")	blcmd_vc1_poweron blcmd vc1.poweron

It is obviously that the hardware specific part VCP is eliminated from the user functions or macro calls. Therefore any device with similar options maybe exchanged without big incidence for the user.
The argument structure may appear rather complicated but the macro package derived functions offering a verbose interface to the user as well as a quiet device interface for further macro programming at the same time. Furthermore increases the clear and straight forward command structure for any implemented device the user acceptance and comprehension.
In case of reading a certain property the user may type blread_vc1_XYZ(1) or blread vc1.XYZ to get a print out of the current value. Otherwise any macro may use blread_vc1_XYZ([0]) to obtain the value of the property returned silently. The 0 is optionally because a not set argument is implicitly set as 0.

On the other hand blset_vc1_XYZ(1,3) sets the value 3 silently to the device property and the use of the argumentless version blset_vc1_XYZ() indicates the request for a user dialog. The macro versions of the same functions are blset vc1.XYZ 3 and blset vc1.XYZ respectively.

DEVICE EXPORT

SPEC supports among other things the export of variables and arrays and furthermore the remote execution of code by a socket connection. This server functionality is well developed but isn't SPEC's main goal. Some care is advisable concerning the bandwidth of a single socket connection and therefore the strategy for data exchange influences the benefit.
The internal structure of the devices organized by the macro package, as stated before, is concentrated in two arrays which stores the basic set of information about a device. The name and the name of the device class array can be obtained and therefore the whole information set maybe derived in a second step. Observing and exporting these two arrays into a client application offers the option to derive the complete state information about all macro package managed devices if additionally the device class arrays are obtained as well.
This approach minimizes the total number of variables to be observed by the client and the run-time influence of the steady client-server connection. Only 2 + N variables needs to be tracked.
The realized client itself is designed as a TANGO server. The first one offers a generic access to all macro package devices too.
Due to the strict data organization the TANGO-server can offer a generic and complete interface to access any property for reading and writing (if applicable). The generic functions are string based and schematically (the original TANGO calls are a bit less instructive) defined as follows:

- string SPECgetdevices();
- string SPECgetproperties(string device);
- string SPECblread(string device, string property);
- void SPECblset(string device, string property , string value);
- void SPECblcmd(string device, string command);

All reading interface functions are operating with a buffered and automatically updated data base. Settings and commands are scheduled into SPEC's command queue

CONCLUSIONS

The introduced macro package in combination with a few naming rules offers a straight forward approach to an object like access for device implementations with SPEC's macro language. Unwanted variable cross talking is maximally avoided and a systematic macro generated user interface can be provided at the same time.

The whole functionality can be exported into a socket client which offers itself a TANGO server for the SPEC macro package managed devices and permits a remote control of them by other programs

REFERENCES

[1] http://www.certif.com
[2] http://www.tango-controls.org

STUDY CASE OF A COLLABORATION PORTAL FOR A INTERNATIONAL SCIENTIFIC PROJECT

Marcin Trycz and Luciano Catani, INFN-Roma Tor Vergata,
Via della Ricerca Scientifica, 1 - Roma, Italy

Abstract

In this paper we present the results of the design, development and preliminary evaluation tests of a web-based collaboration portal aimed at supporting the teamwork of an international scientific collaboration.

In the academic research environment often people use very simple collaboration tools, usually chosen out of habit. In the case of international collaborative projects, in which people don't work physically in the same place for most of the time, these important tools are far from being effective and appropriate. For instance, a collaborative scientific project is made of teams of specialists from different research institutions and countries that need to share files, drawings, pictures, software etc. and document the progress of their work. The different tasks of the project are managed by work groups (WGs) of specialists that organize their work by scheduling meetings, workshops and by setting deadlines. Quite often a single researcher contributes to more than one work group.

The aim of our Portal is to offer a suite of web instruments fulfilling the above requirements without adding extra complexity to the procedures the scientists are familiar with.

INTRODUCTION

In the present-day world of science, research projects are often carried out by large collaborations of different research institutions and universities. The need of specialists for each task of the project requires the contribution of top-quality scientists from different parts of the world working in collaboration across the different phases of the project development: design, operation and the analysis of results.

Sometime the collaboration is based on in-kind contribution from each partner requiring a constant interaction to ensure the perfect matching of the components.

Given these requirements, a continuous and effective communication among members of work groups, and a constant coordination of the latter, is crucial for the successful development of the tasks. At higher level, WGs leaders should continuously check the progress of their own group against each other to ensure a uniform development of the project.

Scientists, compared to many other professional communities, are certainly skilled and well trained in using computers and computer networks because of the important role these instruments have in their daily work.

As consequence of this familiarity, scientists spontaneously tend to profit from computer based communication and collaboration tools, selecting by themselves the solution they consider more appropriated.

This explains the tendency to develop solutions to their collaboration needs that simply implement the tools they are more familiar with: email especially, for communication and documents distribution, file servers, Internet shared agenda, polling services etc.

Experience teaches that, in spite of their familiarity with Internet technologies, or probably as a consequence of it, scientists are somehow reluctant to accept dedicated all-in-one project management solutions that might be selected and suggested by the management. Often, they are convinced that the collaboration instruments they currently use are sufficient or even more effective that the new one.

The above considerations suggested us to start the development of a web portal aimed to providing a "smooth" replacement of basic communication and collaboration tools with a centralized server.

OVERVIEW OF TOOLS AND TECHNOLOGIES

The first task of our analysis process was the identification of the framework suited for our needs.

We ended up with three candidates representing a wider spectrum of technologies: Xoops, Joomla and Liferay. The first two are PHP frameworks; the latter is by now the only open source Java Portal. A deeper analysis showed that only Liferay would have fulfilled our user scheme. On the other hand this portal it's far from being simple, but its complexity can easily be hidden to the final user.

THE COLLABORATION PORTAL

The Portal is meant to be a web-based integrated set of tools supporting a large collaborative project as a communication and documentation service. The two main goals of the Portal are prompt and effective information sharing and well ordered archiving. The Portal also contains additional services that support other aspects of the collaborative work, a calendar for instance, and by taking advantage of the Java portlet [1] “plugability”, others can be added if need be.

Users Management

As we already mentioned, the main reason for adopting Liferay was its powerful user management capability. In international, or large national scientific collaboration scientists from many different research organizations

contribute to the development and operation of a large device, experiment, facility by collaborating to the progress of the different activities dealing with the tasks of the project.

Work Groups, constituted by experts of different research organizations, are created around each task. Typically, scientists from each organization are distributed in many of the project's WGs and every single scientist might contribute to more than one WG. The collaborative Portal should take into account this organization for a targeted and effective deployment of it services (Fig.1).

In Liferay a user's profile can be defined in details and used to configure the services, roles, privileges and even the layout of the user's home page with the information and instruments relevant for role and responsibilities of that particular user. Users won't need to browse the whole Portal to find what they're looking for; sections that are not interesting, or forbidden, will be hidden.

This powerful management of users profile allows creating easily and effectively the different workspaces of the Portal. Work Groups homepage (Fig.2) are accessible only to members of that particular WG and links to these WG homepages are automatically available in the homepage of their members. Similarly, Institutions homepages are accessible only to users affiliated to that research organization.

As final result the Portal is automatically customized for each user according to its profile. Moreover, since users will find their own environment at the first login, learning effort will be very limited.

Portal Workspaces

One of the main concepts and functional components of the Portal are the workspaces. Liferay allows for much flexibility in this aspect, the workgroups, the institutions, even single users can have their own separate workspace. The administrators can decide if the contents of a workspace need to be accessible for reading to all the registered users (i.e. all participants to the project) or only to members of that particular Workgroup.

The management of the Workgroups is dynamic. It means that the project structure doesn't need to be set a priori when the Portal is under development; the Workgroups can be added and removed any time when the Portal is running.

Although Liferay offers other powerful features, for our purposes we chose to use only part of its structure. For instance we didn't implement user's personal workspaces to simplify the interaction with the Portal and to focus user's attention to collaborative activities and services.

The access to the Portal is restricted. Nevertheless the Administrator can configure the workspaces in such way to make some information accessible to unregistered users.

Applications and Tools

The Logbook and the Document Library are the core of the Portal's functionality. The Logbook, developed by customizing the original Blog portlet, is the main tool for sharing information and documenting the progress of the teamwork. The Document Library is the repository for both files uploaded by means of either the built-in interface or as attachments to Logbook entries. While editing Logbook entries, attachments can be assigned to a specific folder according to the topic, achieving a well-organized allocation of files in the repository, which can be accessed via either the Logbook's interface or a dedicated browsing page.

The Calendar allows management of events and deadlines; users can be reminded of relevant events through an automated email system. The Bookmarks tool allows highlighting important links in a side frame. The Activities portlet aggregates all the relevant recent activities, sorted by date, at a glance: new Logbook entries, recent uploaded documents and new Calendar events.

Search is enabled for all of the Portal's content, and is implemented with Lucene text-search library.

As already mentioned before, the main feature of the Portal is the integration of these simple, well-known tools into a fully functional application.

The inter-operation and the co-presence of these tools in a managed environment provide added value to this solution.

As an example, we present in more detail benefits offered by the Portal for some collaborative tasks as compared to the "standard solutions":

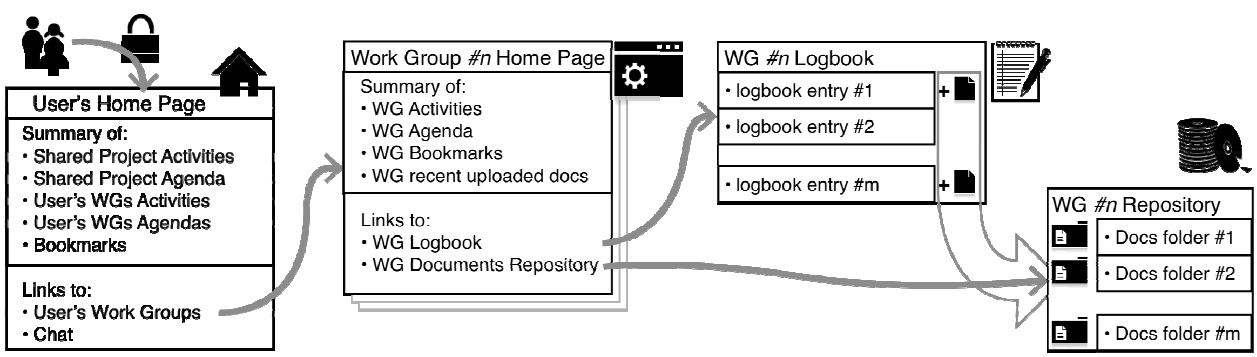


Figure 1: Basic navigation workflow

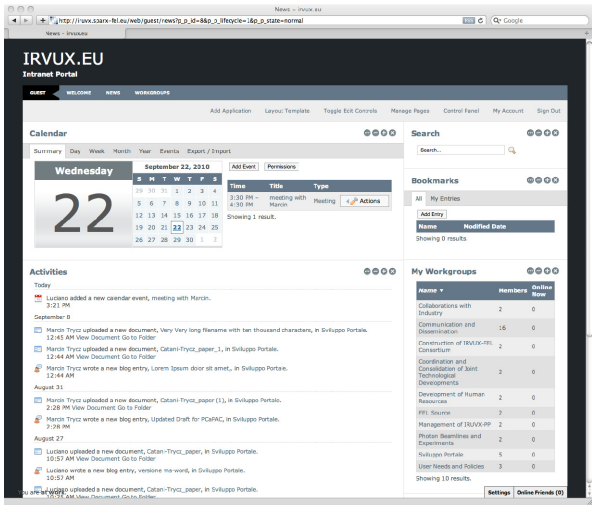


Figure 2: An example of user's home page.

Documenting a work group activities

This duty is usually accomplished by sending email to all WG members and each user needs to organize its own mailing list in the mail browser and keep it up to date. A typical issue is tracking back the old messages in the search for particular information. That is usually not trivial, although some mail browsers allow threading by subject. The Logbook (Fig.3) approach is still simple but more effective: entries are available to all current members of the WG in a dedicated section of the Portal, sorted by date and stored for later browsing. An RSS service, available for this as well as for many other sections of the Portal, allows users to be promptly informed about relevant activities.

Files distributing and archiving

The dedicated Documents Library is a more efficient replacement of a file server.

Figure 3: The Logbook page

A WG can choose a simple one-level folders structure, another might need a more complex multi-level tree-like file system for a structured archiving. The main way to add files to the Library is through the Logbook interface, but single files can be added through the dedicated Documents Library interface. Either way a message that a new file has been uploaded to the Portal will be added in the Recent Activities section in both user's and WGs' homepage. As well as the Logbook entries, all the files can be searched by title and by content in full-text mode, given it's a text-based file like txt, doc or pdf.

Setting up a meeting, deadline, reminder etc.

Any kind of event relevant to the Project or WGs' life can easily be added through the Calendar's advanced interface. Many different event types can be managed: recurring events, multi-day events, etc. The events are time zoned by default, simplifying international users access. Furthermore, all the users potentially interested to the event will receive an automated remainder via email before it's beginning, with customizable advance time.

FIRST ADOPTION

The European IRUVX* collaboration has been the first real-life adopter of the Portal. IRUVX project perfectly fits with the target of the Collaboration Portal: it's an international collaboration aimed to the development of an international consortium of FEL facilities; it addresses different tasks that are managed by working groups in a coordinated effort.

During the test period members of the collaboration involved in the evaluation actively helped us by debugging the Portal services and also suggested minor tweaks for usability. With this experience we ended up with a fine-tuned working Portal, confirmed by the good user satisfaction.

CONCLUSION

The development of a web-based collaboration portal aimed at supporting the teamwork of an international scientific collaboration has been completed. The Portal has been under test for several months by an international scientific collaboration and results confirmed the effectiveness of the collaborative services it provides. Modularity of the framework allows to easily customize and expand the services to any particular user needs.

REFERENCES

[1] http://www.liferay.com/.

[2] A. Abdelnur and S. Hepper, "JavaTM Portlet Specification v.1.0"; http://jcp.org/aboutJava/communityprocess/final/jsr168/

* The preparatory phase of EuroFEL (IRUVX-PP) is funded by the European Commission under FP7.

DEVELOPMENT OF IMAGE PROCESSING SYSTEM ON EMBEDDED EPICS FOR BEAM DIAGNOSTICS

J. Odagiri, K. Furukawa, T. Obina, M. Satoh, High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki, Japan

Abstract

A new image processing system was developed based on EPICS and the FA-M3 PLC made by Yokogawa Electric Corporation. The hardware of the system comprises an F3RP61 CPU module running Linux and an F3UM02 frame grabber module. The CPU functions as an IOC to analyze the raw image data acquired and transferred by the frame grabber module on the PCI-bus, which connects the two modules. A custom record, graphicsRecord, holds the raw image data and the results of analysis as well as parameters set by the user over the network. GUI panels were created by using EDM in order to display the image and to set relevant control parameters into the fields of the graphicsRecord being stationed on the memory of the F3RP61-based IOC. It was confirmed that the developed system is able to acquire image data, analyze them appropriately, and send them over the network to a host computer to display the results of analysis. The design and results on performance measurement of the system is also reported.

INTRODUCTION

It had been common practice to use a desktop PC with frame grabber boards installed in it for beam profile monitoring. This approach allows us to broaden the range of choice of the frame grabber boards and the PC for the purpose. On the other hand, short lifetime of the products and less reliability of the hardware forces us to replace the system frequently to increase burden in maintaining the system in the long run.

In order to solve the problem, we have adopted embedded technology with Experimental and Industrial Control System (EPICS) running on a Programmable Logic Controller (PLC) made by Yokogawa Electric Corporation [1]. Fig. 1 shows the image processing system under test. The main specifications of the F3UM02 frame grabber module are listed in Table 1.

HARDWARE CONFIGURATION

The system comprises an F3RP61 CPU, which runs Linux as its Operating System (OS), and an F3UM02 frame grabber module. The two modules are connected with each other by using not only the PLC-bus on the backplane but also an additional PCI-bus. Both of the modules have a PCI- connector on the side panel to stack them for faster data transfer. The image data acquired with the frame grabber module is transferred to the F3RP61-based CPU by using DMA. The CPU executes the Input / Output Controller (IOC) core program of EPICS on Linux. The IOC analyzes the raw image data and sends it with analyzed results to a host computer which functions as an Operator Interface (OPI) of EPICS.

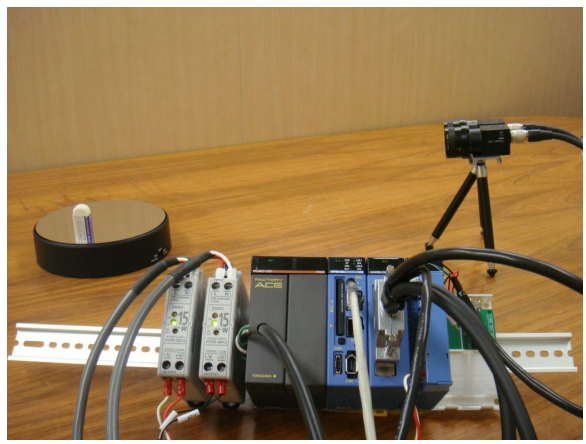


Figure 1: Image processing PLC unit under test. The left most black module (two slots) is the power supply module. The F3RP61 CPU comes to the right of the power supply module. The module just right to the CPU module is the F3UM02 frame grabber module.

Table 1: Main Specifications of F3UM02

Item	Specification
Number of Channels	2ch
Compatible Camera Types	Single Tap (8bits/pixel) Dual Tap (bits/pixel) RGB Colour (24bits/pixel)
Max. Connections	6 monochrome cameras
Resolution of Digitizer/Channel	8 bits
A/D Converter Frequency	100 MHz

SOFTWARE DEVELOPMENT

Record Support

An existing spherical record type, graphicsRecord, which had been created for a seat-gas beam profile monitor was used with some modifications for the analysis of raw image data, such as subtraction of background image, calculation of the projection to both horizontal and vertical directions, searching the peak position in the projection, calculation of the total amount of the light and so forth [2].

Device Support

A new device support module was developed in order to interface graphicsRecord with the hardware. The device support makes the instance of graphicsRecord be processed upon every acquisition of a new image frame by issuing an "I/O_interrupt" scan request. What the device support does is just to transfer the raw image data from the hardware into the buffer of an instance of graphicsRecord. All the other processing of raw image data is subject to the graphicsRecord module.

Operator Interface

Extensible Display Manager (EDM) [3] was chosen for developing the Graphical User Interface (GUI) of the image processing system since it has a type of object which can display an array of data in the form of a two dimensional array of arrays. The feature enables us to display image on the GUI panel from one dimensional array of data stored in the buffer of a graphicsRecord instance as shown in Fig. 2.

TEST OF BASIC FUNCTIONS

To confirm that the device and record support modules function as expected, we have tested the system with a simple object. (See, Fig. 2). The result showed that:

- Captured image was successfully transferred from hardware to the buffer of an snstance of the graphicsRecord.
- Image analysis, such as, creation of projection to both horizontal and vertical directions, peak search, subtraction of background (See, Fig. 3) were successfully executed with the graphicsRecord module.
- The raw image and analyzed results were successfully transferred to the host computer to display them on the EDM-based panel.

All the monitoring and control operations were done via Channel Access (CA) of EPICS which connects F3RP61-based IOC and the host computer over the network.

PERFORMANCE MEASUREMENT

In such a system like PLC, where hardware resource is rather limited, a performance can be an issue. The performance measurement was also done by monitoring CPU power consumption with running the system with various different conditions. The CPU loads measured when no image analysis and no channel access activities were listed in Table 2. Table 3 and Table 4 list the CPU loads measured in case only one of the analysis of raw image or the CA activity between the F3RP61-based IOC and the host computer was in execution. In this measurement, the frame grabber module, F3UM02, was running in external trigger mode and a DC output module was used as the trigger source. While all the tables are subject to a case where one channel of image is being acquired, we have confirmed that the results scale with the number of channels by using two cameras.

More detailed tests revealed that creating projection data costs a lot more than other analysis and making it the most part of the cause of CPU power consumption.

Table 2: CPU Power Consumption
(No Analysis, No Channel Access)

Repetition Period	CPU Load (Typical)	CPU Load (Max.)
1 second	3.00 %	4.00 %
0.5 second	3.70 %	7.30 %
0.2 second	16.0 %	17.0 %
0.1 second	31.0 %	32.6 %

Table 3: CPU Power Consumption
(Only Analysis)

Repetition Period	CPU Load (Typical)	CPU Load (Max.)
1 second	18.6 %	19.0 %
0.5 second	37.0 %	37.3 %
0.2 second	91.3 %	91.9 %
0.1 second	N.A.	N.A.

Table 4: CPU Power Consumption
(Only Channel Access)

Repetition Period	CPU Load (Typical)	CPU Load (Max.)
1 second	6.70 %	7.30 %
0.5 second	13.7 %	14.0 %
0.2 second	34.0 %	35.0 %
0.1 second	69.0 %	70.0 %

SUMMARY

A new image processing system was developed based on an embedded EPICS technology by using a PLC's CPU which executes Linux as its OS and a frame grabber module of the PLC. A special record, graphicsRecord, was ported onto the F3RP61-based IOC and a new device support was developed to interface the record with the hardware. The test results of the system showed that the developed software woks as expected. The result of performance measurement showed that creating projection data is the most part of the cause of CPU power consumption and gives the limit of the repetition rate of image analysis or the number of channels of image data which the developed system can handle.

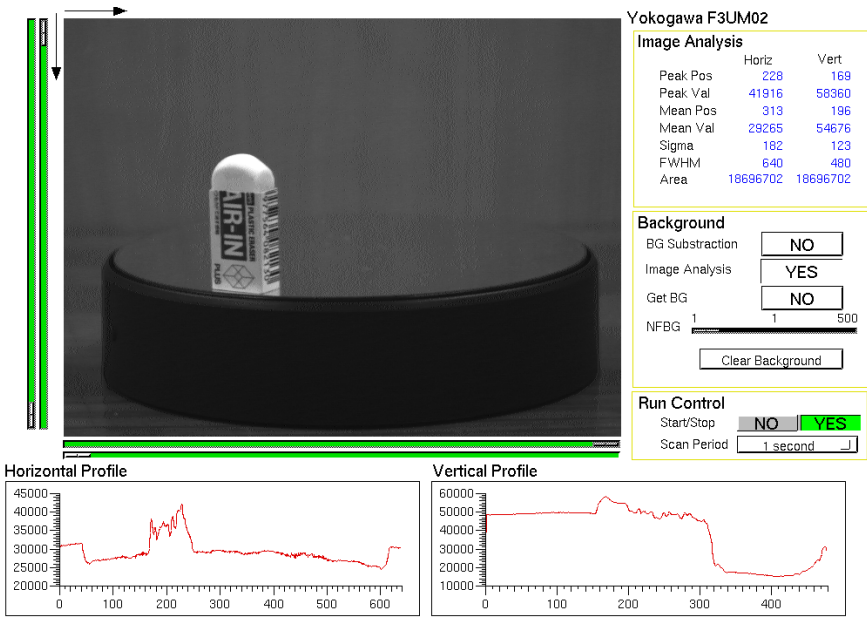


Figure 2: EDM-based graphical user interface. The numbers and buttons at the right side of the image shows the results of analysis and control channels respectively. Horizontal and vertical profiles are shown in the lower part of the GUI.

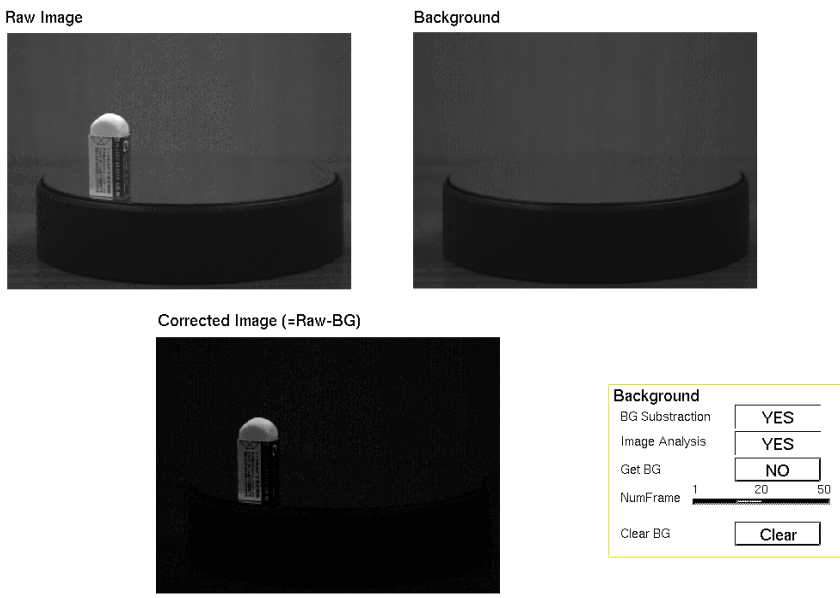


Figure 3: Subtraction of background from raw image. The left side of the upper image shows the raw data. The right side of the upper image shows background data. The result of subtraction is shown in the lower image.

REFERENCES

[1] J. Odagiri et al., "Application of EPICS on F3RP61 to Accelerator Control", Proc. of the 2009 International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2009), Kobe, Japan, Oct. 12-16, 2009.

[2] Y. Yuasa et al., "A Monitoring System for a Gas-sheet Beam Profile Monitor on Linux with EPICS", Proc. of the 2003 International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2003)), Gyeongiu, Korea, Oct. 13-17, 2003.

[3] http://www.aps.anl.gov/epics/docs/USPAS2007/lectures/EDM.odp

CONTROL AND ACQUISITION SOFTWARE COMPLEX FOR TBTS EXPERIMENTS

A. Dubrovskiy, CERN, Geneva, Switzerland

Abstract

The Two-beam Test-stand (TBTS) is a test area in the CLIC Test Facility (CTF3) to demonstrate the high power RF extraction and acceleration at a high accelerating gradient, which are feasibility issues for the Compact Linear Collider (CLIC) project. In order to achieve an efficient data collection, an acquisition and logging software system was developed. All year round these systems store the main parameters such as beam position, beam current, vacuum level, pulse length etc. For predefined events they also gather and store all information about the last several pulses and the machine status. A GUI interface allows from anywhere to plot many logged characteristics at a maximum of 10 minutes delay, to go though all events and to extract any logged data. A control interface configures actions and long-term control procedures for conditioning accelerating structures. The flexible configuration of the logging, the acquisition and the control systems are integrated into the same GUI. After two years operation the critical components have shown highly fault-tolerant. Logging data are used for physic researches.

INTRODUCTION

CTF3 is a test facility which addresses the feasibility demonstration of the Compact Linear Collider (CLIC) [1,2]. The CLIC machine will produce electron-positron collisions at the nominal center of mass energy of 3 TeV at a luminosity of $2\times10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ with a two-beam acceleration scheme. This scheme is studied in the Two-beam Test-stand (TBTS), which is a part of CTF3. An electron beam (the drive beam) of 12 GHz is generated from a 1.5 GHz electron beam in a Delay Loop and Combiner Ring and then sent to the TBTS. The drive beam of an intensity of up to 32 Amps passes through a Power Extraction Structure (PETS). The extracted 12 GHz RF power from the drive beam is used to accelerate the second, low-intensity, beam (the probe beam). In the TBTS set-up the CLIC feasibility, stability and protection issues are studied, such as the beam changes during the deceleration, the RF extraction properties by the PETS, the high-gradient acceleration, as well as the Two-beam scheme performance and the fault-tolerance [3].

A control and acquisition high-level software complex was developed in order to assist all TBTS experiments, measurements and control routines. From the user point of view, the acquisition and logging parts of the system must be extremely reliable and robust; and it must work round-the-clock. The software system is flexible and adaptive to failures of hardware or software components involved in the TBTS set-up. Another issue is that the development of the software continues during several years such that it follows the requirements of R&D experiments and the hardware installation; and it remains light in support and compatible. The automatic control part contains a material protection mechanism and an accelerating structure processing.

FRAMEWORK

The TBTS software design approach is based on a model-driven architecture. The software developing process contains two distinct periods of time. During the first and initial period the developer followed the waterfall model approach. Specifications for different software aspects were completed iteratively during an extended period of time. That is why the first four stages of the software development consecutively alternated: requirements analysis → software design → integration → testing → requirements analysis → and so on. During this period the full range testing is very time consuming and some aspects remain unknown. Hence the testing, the validation and the performance estimations were made for some aspects of typical situations. The model merging is one of the most difficult processes during this phase. In order to simplify this problem, a core model was designed, which covers the static part of the set-up and it remains independent of the software and hardware realisations. The core software model was developed based on the instrumentation, controllers and machine time triggers layouts and general specifications. The acquisition model defines the generalized device interfaces for different data access interfaces and different types of equipments. The control model depends on only the core model and the acquisition model. At the end of the first period most of this was defined and realized in the server part of the software. The remaining part is gradually put in operation during the second period taking into account the importance of the blocks. So the second period of the development relies on the feed-back from results, goals and tasks of experiments and set-up changes. During this period the development becomes lighter and faster, the development tends to be agile.

ACQUISITION

The acquisition part of the software complex is to obtain all necessary information about the CTF3 machine status and the experiment. CTF3 can run in several modes for the TBTS beam lines:

- only the drive beam is on;
- only the probe beam is on; synchronizly;
- probe and drive beams are on, but not synchronized;
- probe and drive beams are on and synchronized.

Moreover all measurement equipments are located on different front-end crates in the network, and

measurements are updated non-simultaneously. Thus in all running modes the acquired data must be synchronized in a way to get the beam, RF and other measurements resolved for each pulse in both beam lines. If the read-out of some of equipments fails, the missing information is treated as a special case.

Machine status

In CTF3 there are several parameters which relevant to TBTS beam properties. The status of the electron gun and the pulse length define the initial beam. Meanwhile the recombined beam pulse can be cut by a tail clipper, which is installed after the extraction from the Combiner Ring. The CTF3 safety interlock system protects the equipment and the personal from harm. Any TBTS control activity is stopped on the activation of one of interlocks; and it resumes when the system is okay. All these parameters are synchronized with a CTF3 acquisition trigger and they are synchronously acquired by the TBTS software.

At present the distribution in the waveguide system of the extracted RF from the PETS is controlled by RF actuators: two RF attenuators and one RF phase shifter that can be remotely changed. The actual position of stepping motors is read using a spring return. The read-out and the control of RF actuators are also synchronized with CTF3 triggers.

RF Simulation

Most of the time the TBTS set-up is running in the RF recirculation mode: a part of the extracted RF feeds back the PETS with a certain phase shift and the other part of RF goes to the accelerating structure [4]. Taking into account the status of RF actuators, the RF propagation in the RF waveguide system is simulated based on the drive beam intensity for every pulse. The simulation is compared to the RF power measured by directional couplers, which are installed in 5 different places. This allows detecting anomalies of the RF transportation and recirculation every pulse. In case of a normal pulse the comparison between the simulation and the measurement gives the beam quality: the bunch form factor and the beam phase along the pulse.

Pulse Summary

Every pulse, the full set of data is summarized into a set of scalar values. The summarized data is used later in the control part of the software. For the user it is a possibility to monitor the evolution and the processing of the system. The main waveform measurements are summarized into the data set by the type of measurement:

- forward power – the peak power, the total power over the pulse duration, and the total period, when the power exceeds 50%, 75% and 90% of the peak power;
- reflected power - the peak power and the total power over the pulse duration;
- BPM – the mean current, vertical and horizontal positions at the flat top of the intensity waveform;
- Faraday cup – the peak signal.

Based on the predefined conditions the acquisition system determines anomalies during the high-field travelling in three sections of the TBTS: in the RF recirculation loop, in the RF waveguides towards the probe beam and in the accelerating structure. The typical indications are a high relative reflection, a missing energy, a high ion emission and a significant difference between the measured and the simulated RF.

Event

In order to minimise the amount of logged data and to provide a “one row data access”, the event system was implemented. For predefined conditions, such as breakdowns, interlocks and errors, the event system gathers all data together about the last and several preceding pulses and the machine status. In particular data from about 50 additional signals predefined by the user and two MTV cameras are acquired and saved, which are analysed off-line. The user can also raise an event by an external trigger from the GUI panel or he can activate the periodical event trigger.

CONTROL

The software system controls relevant TBTS actuators: the gun mode, the gun interlock, the gun pulse length, the tail clipper, the RF attenuators and the RF phase shifter. The user can control actuators in physical units. Control subsystems were implemented that atomise the control routine. The most important controllers are the interlock and the accelerating structure conditioning systems.

Interlocks

The interlock control subsystem is needed to protect the experiment hardware and to provide the purity of the experiment. There are two typical actions on an interlock: to cease the drive beam and to reduce the power production. These actions avoid problems with high vacuum, beam losses, klystron instabilities, PETS and accelerating structure breakdowns. The main indications for an interlock are a high reflection, a high vacuum level, beam losses, a missing energy and a breakdown. After vacuum level sparks the system waits until the vacuum level is below the normal level. Similarly, the system waits a predefined time to calm down the experiment set-up after a breakdown. The operation is automatically resumed only when all detected problems are solved.

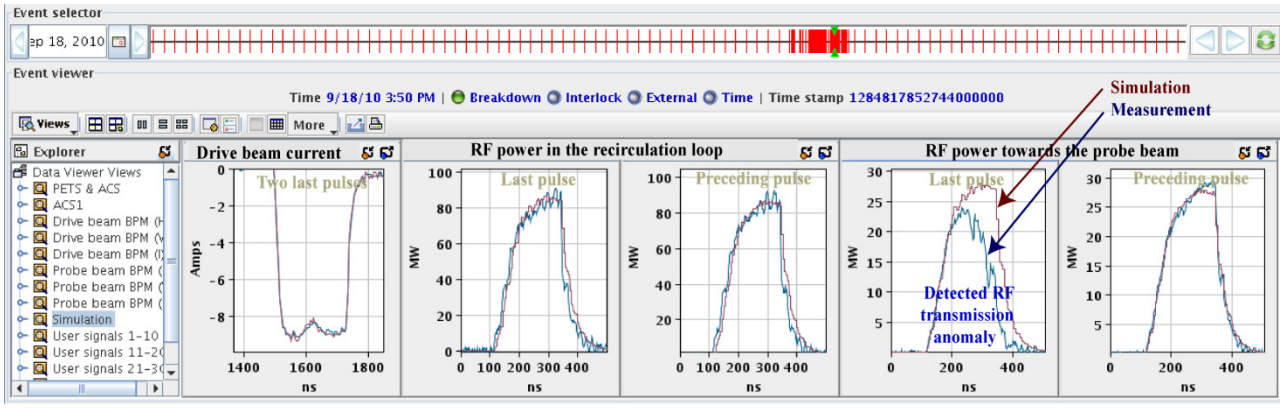


Figure 1: Illustration of the event chronograph, where a detected RF transmission anomaly in the waveguide is shown.

Conditioning controls

The conditioning strategy is an automatization of the conditioning, preparation and the long-term measurement processes. The control software must initialise the machine with an initial attenuator position and the pulse length and verify that all interlocks are deactivated. Then the programme ramps up the pulse length to a target pulse length by a pulse length step and a delay between changes. After that it changes the attenuator stepping motor position towards the target attenuation by steps and delays. If a breakdown occurs during that procedure, the control system must follow the interlock specification. If the number of breakdowns exceeds a threshold over a period, the programme should increase the target attenuation by a specified increment. If the procedure reaches the target positions and it stays for more than a predefined time, the programme reduces the target attenuation by a certain decrement. All parameters of the conditioning strategy are defined by the user.

LOGGING & GUI

The logging system permanently stores most of the acquisition data and many other parameters, in total several thousand parameters. All data are available after a several seconds for CERN internal users and after less than 10 minutes for external users.

The GUI part is composed of different display instruments. Quick access panels are an actuator control panel, an experiment description, the last pulse summary and the logging status. Remote configurations are an accelerating structure processing setup, interlock configurations, connections and signal treating configurations, RF simulation settings and others. The logging tools are a logging data plotting and the data extraction into MAT-format files. Acquisition visualizations are a last pulse waveform display and an event chronograph display. An example of the event chronograph display is shown in Fig.1, where a RF transmission anomaly was automatically detected.

CONCLUSIONS

An approach to develop dedicated software for the TBTS research was worked out. It allowed to create a model based system with rich functionality and flexibility, which meets the physics requirements. It is light in support during the operation. The complex of developed systems has been used in CTF3 for two years. The software has shown highly fault-tolerant and it is an efficient instrument within the scope of Two-Beam studies.

ACKNOWLEDGEMENTS

The author acknowledges F. Tecker for useful discussions, comments and the careful correction of this work. Also the author thanks to E. Adli, I. Syratchev, R. Ruber and S. Doebert for comments and explanations.

REFERENCES

[1] J.P. Delahaye, “Towards CLIC Feasibility”,IPAC’10, Kyoto (2010), FRXCMH01.

[2] P.K. Skowronski et al., “Progress Towards the CLIC Feasibility Demonstration in CTF3”, IPAC’10, Kyoto (2010), WEPE027.

[3] R.J.M.Y. Ruber et al., “The CTF3 Two-beam Test-stand Installation and Experimental Program”, EPAC’08 (2008), WEPP139.

[4] E. Adli et al., “First Beam Tests of the CLIC Power Extraction Structure with the Two-Beam Test Stand”, DIPAC'09, Basel (2009), MOPD29.

ESTIMATION OF THE RESPONSE TIME AND DATA FLOWS IN THE TOTEM DETECTOR CONTROL SYSTEM

F. Lucas Rodríguez, CERN, Geneva, Switzerland

Abstract

The TOTEM experiment at the LHC is composed by 3 different detectors, (Roman Pots silicon detectors, CSC T1 and GEM T2 telescopes). The Detector Control System (DCS) is generated in a highly automated process from external representations for connectivity and behavior. From these representations (one of them, the Finite State Machine tree) it is also possible to estimate the response of the system. It is possible to assign weights to each one of the nodes and estimate data transfers among subsystems, memory consumptions, reaction times, storage needs,... The main purpose of those estimations is not to do a full predictability analysis of the system, but just to provide a help in case of performance problems.

INTRODUCTION

The TOTEM (Total crOss secTion, Elastic scattering and diffraction dissociation Measurements) experiment at CERN [1, 2] will measure the size of the proton and also monitor accurately the LHCs luminosity [3]. To do this TOTEM must be able to detect particles produced very close to the LHC beams.

TOTEM consists of "Roman Pot Stations" (RP), "Cathode Strip Chambers" (CSC) Telescope 1 (T1) and "Gas Electron Multipliers" (GEM) Telescope 2 (T2). The T1 and T2 detectors are located on each side of the CMS interaction point in the very forward region, but still within the CMS cavern. Two Roman Pot stations are located on each side of the interaction point at 220 m and 147 m inside the LHC tunnel. Each Roman Pot station consists of two groups of three Roman Pots separated by a few meters.

Such kind is in the learning phase that will produce elaborated requirements for the Control System [4] [5].

TOOL FOR THE CALCULATIONS

An specific tool has been developed for the calculation of the rate of information exchanged among all the hardware component using the method proposed in next Section. It uses the pinout information of the detector and some heuristics to build a Finite State Machine (FSM) of the detector [6]. Also assigns to each level different values according to the Product Breakdown Structure (PBS) tag for the calculation factors as seen in Table 1.

Table 1: PBS configuration entry for Temp. Sensors

Property	Value	Explanation
Id	E.03.05.03	The identifier in the PBS.
Description	E. M. - Temp.	A text description of the PBS identifier.
Information Chunk	4.00 B	The information size of the value transmitted in the readout.
Variation Prob.	1	The probability of changing the value between two readout intervals.
Archiving Freq.	300 s	This value is multiplied by the probability of change and the information size.
Archiving Node	4.70 GB	Is the information that this node has to archive only by itself.
Archiving Overhead	16.00 B	Represents the overhead in the structure to store the InformationChunk in a database.
Readout Rate Freq.	500 ms	This value is multiplied by the probability of change and the information size.
Readout Rate Node	37.50 Kb/s	Is the information that this node is generating only by itself.
Readout R. Overhead	8.00 B	Represents the overhead of the InformationChunk in the communication protocol.
Time Execute	100 ms	Time needed to execute a request.

This tool is highly modular, and the process of calculation has three steps:

1. Parse the tables and build the FSM tree.
2. Assign and match the PBS entries in the tree.
3. Execute correspondent algorithm for the calculations.

Each algorithm is contained in a independent class that is dynamically loaded. It explores the FSM tree and the PBS items and calculates certain tags. Also, an algorithm can generate new tags in the PBS item during its execution.

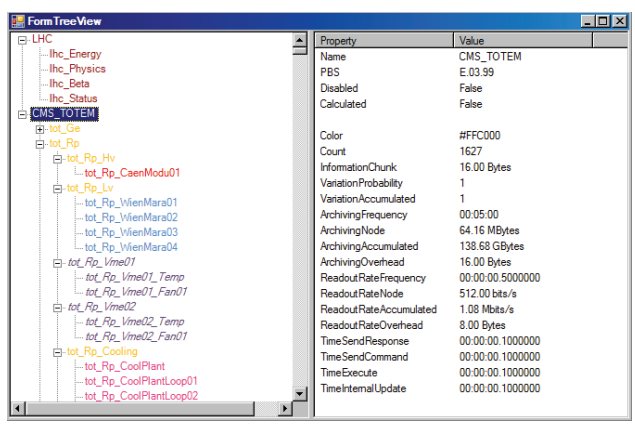


Figure 1: User interface of the tool

ANALYSIS OF THE RESPONSE TIME

Model of the CAN bus: SyncInter Validation

The first step is to calculate the time need to do a full readout of the bus. This is addressed with the *SyncInter* parameter. It indicates with what frequency the ELMB in the CAN bus must be pooled to retrieve the information.

$$\begin{aligned} BaudsInElmbReadout &= NumChannels * FrameLenght* \\ &\quad * BitToBauds\,(\text{baud}) \\ BaudsInCanReadout &= BaudsInElmbReadout \\ &\quad * NumberElmb\,(\text{baud}) \\ AdcInter &= (1/AdcRate) * 1000\,(\text{ms}) \\ ElmbInter &= NumChannels * AdcInter\,(\text{ms}) \end{aligned}$$

So the final value can be expressed such as

$$\begin{aligned} \text{BusExtractInter} &= (BaudsInCanReadout/CanSpeed)* \\ &\quad * 1000\,(\text{baud / (baud/ms)} = \text{ms}) \end{aligned}$$

3 requisites must be satisfied to be a valid *SyncInter*:

- *BusExtractInter must be less than ElmbInter*
  The bus has to be faster than the ELMB generation of data.
- *SyncInter must be bigger than BusExtractInter*
  Is is reasonable to wait the time needed for a full readout an ELMB before triggering a new one. If not the bus utilization would be over 100%.
- *SyncInter must be less than ElmbInter*
  For not loosing measured values it is necessary to trigger and transmit the values faster than they are being generated in the ELMB.

All those 3 requisites can be resumed in the following condition

$$SyncInter \in (BusExtractInter, ElmbInter) \qquad (1)$$

So the bus occupancy can be defined as:

$$Occupancy = (BusExtractInter/SyncInter) * 100$$

Values for the Roman Pots ELMBs chain for Temperature and Vacuum

This is the worst case scenario in TOTEM layout. It is a bus configured at 500000 bauds/s with 4 ELMB with 64 channels configured at a ADC speed of 1.88Hz. The frame length is 29 bits, the increase of the conversion from bit to bauds is factor 1.1.

The interval is (17ms, 34042ms). So an interval of 400 ms fulfills all the requirements and leads to a bus occupancy of 4.1%.

It is important to remark that changing the ADC rate of the ELMB does not affect the bus occupancy. This parameter is only affected by the *SyncInter*. What happens that the change in the ADC rate can lead to a no longer valid *SyncInter* as stated in Equation 1.

Model of the OPC

In the actual TOTEM DCS design, every machine has its own OPC server and client. The HV and LV OPC servers use the concept of OPC groups for refreshing the values.

The two main parameters in a OPC group are:

- *UpdateRate*
  It determines how frequently the client is notified of new values.
- *RefreshTimer*
  If no update is received in this interval, the client forces a poll to the server. Consequently *UpdateRate* must be greater than *RefreshTimer*. If this parameter is 0 this functionality is disabled.

The specification of OPC 3.0 allows to handle OPC items individually without any group. The ELMB OPC server implements this functionality.

Model of the FSM

Each FSM node has a processing time after one of the children changes or after receiving a command for the parent. This time is specific for each node, because it is directly related to the number of children, and the number of states of the node itself. Furthermore it can even include network requests,…

On average a reasonable estimation can be 500 ms for this internal processing time for each node. This value is obtained empirically.

Global chain

The DCS can be considered as a “soft real-time” system. If the sensors are read with some delay, or the commands sent a few milliseconds later, no damage or wrong calculations should take place.

An example of estimation of the response time of a DCS action is the vacuum failure. The DCS needs to react on time by disconnecting the cooling plant loop and the Hv and Lv power. The aim is to avoid condensations with the consecutive formation of ice around the electronics leading to short circuits. This scenario is calculated in *Situation A* of Table 2. The FSM states will propagate upwards 3 levels in the FSM hierarchy up to the proper station. Here a command is sent to the cooling child node to close the loop.

The approach is straightforward:

1. Determine the *RefreshRateSensor* of the vacuum sensor and the actuator. It will be the *SyncInter* of the bus CAN, or the *UpdateRate* for Wiener and CAEN OPC servers.
2. Calculate the *CommonNode* in the FSM hierarchy between the *SensorNode* and the *ActuatorNode*.
3. Assign to *Measure* the number of levels of difference between the *CommonNode* and the sensor.
4. Assign to *Command* the number of levels of difference between the *CommonNode* and the actuator.
5. The total *FsmProcessing* time is (*Measure* + 1 + *Command*) $*$ 500 ms.
6. The final *ReactionTime* for a hypothetical soft interlock is *RefreshRateSensor* + *FsmProcessing* + *RefreshRateActuator*.

Table 2: Soft interlocks estimation

	Situation A	Situation B
Sensor	Vacuum sensor inside a Pot	Temperature sensor inside an Hybrid
Actuator	Cooling circuit for a full station	Wiener Maraton Low Voltage Group
Sensor Node	tot_Rp_45_220_fr_tp_Vacc01	tot_Rp_45_220_fr_tp_Temp01
Actuator Node	tot_Rp_45_220_CoolPlantLoop	tot_Rp_45_220_fr_LvG
Common Node	tot_Rp_45_220	tot_Rp_45_220_fr
Refresh Sensor	400 ms	400 ms
Refresh Actuator	500 ms	3000 ms
Measure	3	2
Command	1	1
FSM Processing	2500 ms	2000 ms
TOTAL *Reaction*	3400 ms	5400 ms

The complementary action of switching off the Hv and Lv is taken inside the Pot level, so the propagation is only 1 level up.

ORDERS OF MAGNITUDE OF THE INFORMATION EXCHANGED

The lifetime of the DCS is estimated to be 20 years of continuous operation. Table 3 gives a total overview of the DCS expected workload considering the RP and T2 detectors.

The main result of this table is that "only" 150 GB are needed for the TOTEM DCS archiving. And the total data-flow of the whole system is slightly above 1 Mbit/s.

The data-flow results are not a CPU load estimation, but are directly related. Even if the data-flow increases up to 2 Mbits/s taking in consideration T1 detector, a usual computer of nowadays could handle the requirements of the whole TOTEM experiment.

Table 3: Extract of distribution from DCS PBS elements

PBS	Description	Count	Archiving	Readout
E.03.01	High Voltage	34	5.33 GB	42.50 Kb/s
E.03.02	Low Voltage	86	8.08 GB	64.50 Kb/s
E.03.03	Front E. Electr.	18	1.13 GB	9.00 Kb/s
E.03.04	DCU	720	67.67 GB	540.00 Kb/s
E.03.05.01	E. M. - Canbus	4	160.40 MB	1.25 Kb/s
E.03.05.02	E. M. - ELMB	11	441.10 MB	3.44 Kb/s
E.03.05.03	E. M. - Temp.	120	4.70 GB	37.50 Kb/s
E.03.05.04	E. M. - Vacuum	144	5.64 GB	45.00 Kb/s
E.03.05.05	E. M. - Humid.	24	1.50 GB	12.00 Kb/s
E.03.06	Cooling	9	1.41 GB	11.25 Kb/s
E.03.08	Motor. values	101	15.82 GB	126.25 Kb/s
E.03.09	D. Safety Sys.	40	6.27 GB	50.00 Kb/s
E.03.99	FSM nodes	308	19.30 GB	154.00 Kb/s
E.03	SUM	1632	**139.47 GB**	**1.09 Mb/s**

CONCLUSIONS

This tool and philosophy provides only an estimation. It cannot be considered as a explicit scheduling mechanism or an study on synchronization.

However it has proved to be extremely useful for the design of the whole system. Usually the DCS is built and then takes place several test to see if it behaves properly. Those experimental test can never be comprehensive enough, and could lead to situations where the timing requirements does not fulfill.

ACKNOWLEDGMENT

I would like to give thanks to the TOTEM collaboration colleagues. They have provided the information needed of how to decompose the system in the different levels and the behaviour implemented in the FSM.

REFERENCES

[1] V. Berardi, M. G. Catanesi *et al.*, "TOTEM: Technical Design Report," CERN-LHCC-2004-002, January 2004.

[2] G. Anelli, G. Antchev *et al.*, "The TOTEM experiment at the CERN Large Hadron Collider," JINST, 2008.

[3] M. Albrow, G. Antchev *et al.*, "Prospects for diffractive and forward physics at the LHC," CERN/LHCC 2006-039/G-124, 2007.

[4] F. Lucas Rodríguez, I. Atanassov *et al.*, "The totem detector control system," in *Proceedings of ICALEPCS 2009*, Kobe, Japan, October 2009.

[5] F. Ravotti, I. Atanassov *et al.*, "The totem on-line radiation monitoring system," in *Proceedings of ICALEPCS 2009*, Kobe, Japan, October 2009.

[6] F. Lucas Rodríguez, I. Atanassov *et al.*, "Automation tools in the software development of the totem detector control system," in *IEEE NSS 2010*, Knoxville, Tennessee, November 2010.

PLANS FOR MONITORING TPS CONTROL SYSTEM INFRASTRUCTURE USING SNMP AND EPICS

Y. T. Chang, Y. K. Chen, Y. S. Cheng, C. Y. Wu, C. H. Kuo, Jenny Chen, C. J. Wang, K. H. Hu, K. T. Hsu

National Synchrotron Radiation Research Center, Hsinchu 30076, Taiwan

Abstract

The Taiwan Photon Source (TPS) control system is one of the crucial systems for the accelerators and beamlines. It is necessary to monitor the status of the control system components such as housekeeping parameters of cPCI EPICS IOC crates, network traffic, connections between computers, etc. The equipment room environment including electric power, temperature, fire alarm, and water leak will also need to be watched. Using Simple Network Management Protocol (SNMP), the behaviour of network-attached devices can be monitored for administrative attention. Since the TPS control system is based upon the EPICS framework, the monitoring system is planned to adopt the EPICS support with SNMP. This paper will describe the system architecture of this monitoring system.

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 2013.

TPS control system will be implemented by using the Experimental Physics and Industrial Control System (EPICS) [2] framework. The various devices are integrated with EPICS based Input Output Controller (IOC) via control network connection. Figure 1 shows the architecture of TPS control system.

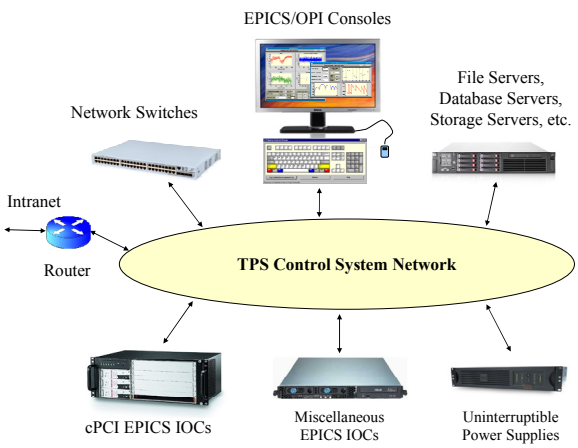


Figure 1: Possible SNMP-compatible devices in the TPS control system network

There are 24 Control Instrumentation Areas (CIA) which distributed along the inner zone just outside of the machine tunnel. Each CIA serves for one cell of the machine control and beamline interface. EPICS IOCs and major control devices connected to the control system are installed inside CIAs.

The TPS control system is designed for high availability. Its infrastructure must be reliable. Due to the long distance between control room and CIAs, an infrastructure monitoring system is planned to be implemented for gathering status of control system components such as CompactPCI (cPCI) IOC crates, network switches, servers, Uninterruptible Power Supplies (UPSs), etc. A dedicated EPICS IOC is planned to be used for housekeeping to monitor the health condition of these devices. When abnormal situations occur, e.g. crate temperature overheat, power supply breakdown, fan failure, or network disconnection, the monitoring system will automatically display the warning messages on the operator interface (OPI) screen and send out the alarm notification by voice call and E-mail. We can receive the early notification before a problem turns into a disaster. In addition to the warning messages, the monitoring system will also generate the warning reports or charts which can indicate the problems at the same time. Software tools such as MATLAB will be used to create these warning reports or charts automatically.

SYSTEM ARCHITECTURE

Simple Network Management Protocol (SNMP) is an industry standard protocol for managing statistical data of the network-attached devices. It is based on the client-server architecture and consists of three components: managed device, agent, and Network Management System (NMS). A managed device is a network node that implements an SNMP interface that allows access to specific information. An agent is a software module that resides on a managed device which reports information via SNMP to the NMS. The NMS is an application which runs on the manager and regularly polls data from agents.

SNMP mechanism associates with the Management Information Bases (MIBs) which describe the structure of the management data of a device. MIB uses a hierarchical namespace containing object identifiers (OID). Each OID identifies a variable that can be read or set via SNMP.

The devSNMP [3] is the EPICS device support with SNMP that allows us to access management data from any network device in the same manner as we are used to for the EPICS PVs. Current devSNMP supports only

snmp-read commands and works with SNMPv2c.

A dedicated EPICS IOC will be used to retrieve information from the SNMP-compatible devices. The system structure is shown in Figure 2. The IOC can query the management data from managed devices via SNMP protocol. Then the data will be stored in the EPICS database for PVs channel access. The Extensible Display Manager (EDM) will be used as the operator interface (OPI) to show the monitored information via the Channel Access (CA) protocol.

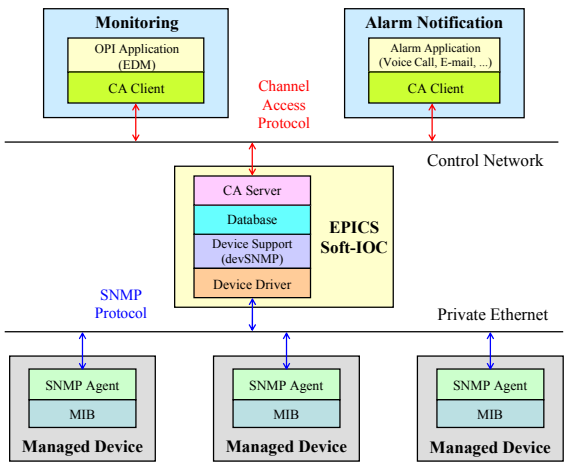


Figure 2: System block diagram of building EPICS support for SNMP-compatible devices

MONITORING OF SNMP-COMPATIBLE DEVICES

In the future, the TPS control system is expected to setup more than 100 IOC crates, more than 50 network switches, and more than 50 UPSs. Due to these large amounts of devices, it is hard to pinpoint which device has a breakdown. Fault-finding and troubleshooting become a critical issue we have to face. So, it is necessary to build a warning mechanism using the SNMP-compatible devices that can show the status of device information on the Graphical User Interface (GUI). Besides the monitoring screen, it can also send the warning message by voice call and E-mail. In order to speed up the troubleshooting, the warning message should indicate the location and status of the fault devices. The detail will be described in the following paragraphs.

cPCI Crates

The cPCI crate is chosen as the standard EPICS IOC platform for the TPS control system. Monitoring the health of the crate is essential. Each cPCI crate has an alarm board with SNMP support. The alarm board can provide the parameters of the crate status. These parameters stand for the status of the following entries, including voltage, temperature, fan speed, and status of power supply unit.

The monitoring system will poll the data from the cPCI crates located at 24 CIAs every 10 seconds. A prototype has been developed to collect the real parameters from cPCI crates for testing. Figure 3 shows the EDM display page for monitoring the cPCI IOC crate status. Each column represents one cPCI IOC crate which will be installed in certain CIA. If the parameters exceed normal range, the display value will turn into red for warning the operator.

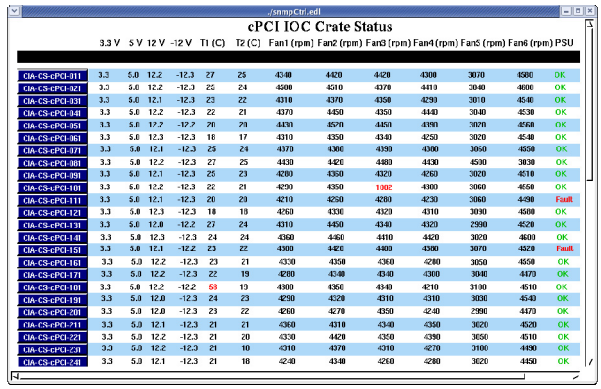


Figure 3: EDM display page for monitoring cPCI IOC crate status

The original devSNMP module should be extended the "Regular Expression" for non-standard output data string. There are two ways to solve the problem: One is to rewrite the device support of devSNMP module, the other is to insert the a new definition field of module support. However the second method could solve the problem much more directly so that the definition "scalcout" from CALC [4] module is inserted to main database definition file. Afterwards, it only need to add the "scalcout" field in the DB file which would translate the output string from the device.

Network Switches

Network Switches only support SNMP, however, they do not support with EPICS framework. We can use off-the-shelf network management tools or dedicated EPICS IOC to obtain the SNMP data of switches.

The application tools for network management (e.g. MRTG, RRDTools, Ganglia, etc) usually have many complicated functions for certain purposes. These tools are suitable for webmaster or network manager who needs to monitor the detail information of network equipments.

In general, most of the users or maintainers only wants to know some ordinary entries such as heartbeat, bandwidth and hot-spot warning. Thus assembling the housekeeping parameters is also needed to make into consideration for monitoring these switches.

In contrast with switching among different kinds of graphical user interfaces, it is more convenient and efficient to centralize variety of data into EPICS IOC so that we can manage and present the received data via a customized control interface which could integrate into the TPS control system.

In Figure 4, it shows that both EPICS IOC with devSNMP module and network management tools are used to monitor the status of switches.

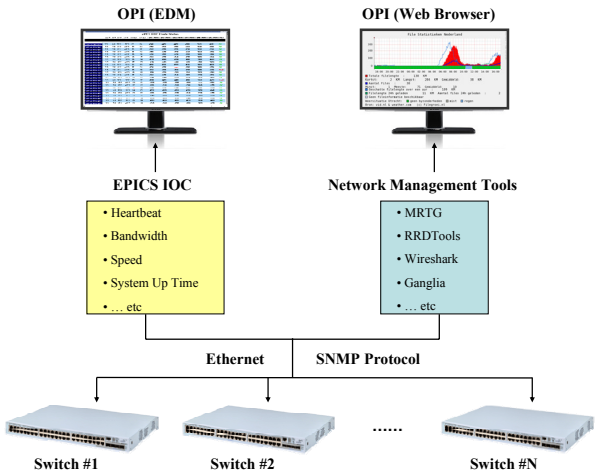


Figure 4: EPICS IOC vs. Network Management Tools

Servers

There are two strategies to monitor the servers, one is used for the servers without running EPICS IOC but support SNMP, the other is for the servers with running EPICS IOC. Figure 5 shows the difference between them.

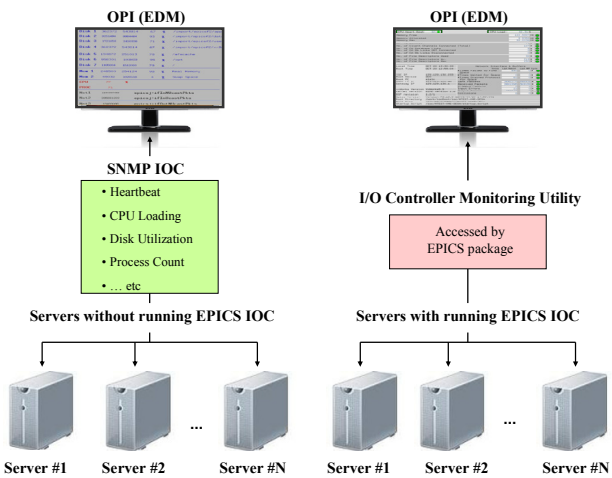


Figure 5: Monitoring schema for servers

For servers without running EPICS IOC, such as file servers and database servers, we can start the SNMP daemon to allow the dedicated EPICS IOC to gather host information. The host information includes heartbeat, CPU load, disk usage, number of processes, network traffic, etc. The housekeeping information such as status of power supply, fan, and temperature can also be obtained via the MIBs provided by vendors.

For servers with running EPICS IOC, there is the IOC monitoring utility similar to IOCMON [5] that can run at IOC and monitor the available resources. The OPI can get data directly through the utility without involving SNMP and other dedicated EPICS IOC. It can reports the IOC resources information including CPU heart beat, CPU usage, number of file descriptors used, memory allocated, boot parameters, number of CA clients, number of CA database links, and network interface statistics, etc.

Others

Other SNMP-compatible devices such as UPSs are planned to be added into the monitoring system. The UPS status information such as current, load rate and battery will be monitored.

The equipment room environment parameters including electric power, temperature, fire alarm, and water leak are also our concerns. Instead of using the inefficient SNMP, detection devices supported by EPICS will be used to collect the environment parameters which can be integrated to the control system.

In order to maintain and classify each device in effective way, EPICS framework supports the template file for management. In particular, "dbLoadTemplate" loads the definition file which contains macros and other substitutions. It is not only reducing the line numbers of the script but also flexible to extend the parameters for each device.

SUMMARY

Maintaining high reliability of the TPS control system is important to the operations of accelerators and beamlines. Since there are many control system components distributed at numerous locations in the TPS buildings, it is necessary to have an infrastructure monitoring system to supervise the status of these components. Most of these components such as IOC crates, network switches, and servers support SNMP which is the industry standard protocol for managing statistical data of the network-attached devices. To be consistent with TPS control system which is based on EPICS framework, the monitoring system is developed by using the EPICS device support with SNMP. This system not only can display warning messages on the OPIs but also send alarm notifications to responsible personnel by voice call and E-mail. The alarm message should contain the location and status information for easily targeting the failed device. A prototype has been developed to gather the real status information from cPCI crates. Implementation for other SNMP-compatible devices is still in progress.

REFERENCES

[1]. TPS Design Book, v16, September 30, 2009.
[2]. Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/
[3]. EPICS SNMP, http://www-mks2.desy.de/content/e4/e40/e41/e12212/index_ger.html
[4]. EPICS calc module, http://www.aps.anl.gov/bcda/synApps/calc/calc.html
[5]. IOCMON : I/O Controller Monitoring Utility, http://epics.web.psi.ch/software/iocmon/

DATA ACQUISITION AND STUDIES OF VIBRATION MOTION IN TLS BEAMLINES

P.C. Chiu, C.H. Kuo, K. H. Hu, Jenny Chen, Y. S. Cheng, Y. K. Chen, K.T. Hsu
NSRRC, Hsinchu 30076, Taiwan

Abstract

TPS (Taiwan Photon Source) is being under construction while TLS (Taiwan Light Source) is still on operation at the same NSRRC site. It was observed that the stability of photon beam intensity (Io) of TLS seemed a little deteriorated at daytime, when civil work is busy, compared to the nighttime. The intensity changes at different beamlines, however, aren't consistent with each other in each time, furthermore not so agreeing with the electron beam. Therefore, to correlate how the ground vibration due to civil construction effected on beam behaviour, the vibration measurement system is integrated into the existing TLS control system. The system will support waveform acquisition which could be acquired on demand. Meanwhile, realtime 10 Hz rms detector which could be archived continuously is also considered to be built in the future.

INTRODUCTION

The TPS is a 3 GeV energy electron ring with 512 meter circumference and planned to be delivered to users' end stations in 2014. During the periods of its constructions, the TLS at the same site will continuous be on operations. The quakes caused by excavators or pile drivers as Fig. 1 seem to have deteriorated the stability of beamline intensity (Δ Io/Io) from 0.1% up to 10% or more. On the other hand, these stability indicators Δ Io/Io between different beamlines have been not always concordant. Furthermore, it has been confused us over a long period that the indicators sometimes became worsen while the related subsystem remained normal even before TPS construction. It is suspected that different characteristics of vibration of different girders quite would be one of possible causes. Therefore, to clarify these inconsistent and not-yet-explained phenomena, the data acquisition system of vibration is planned to be built and continual expanded. In this report, the infrastructure of vibration data acquisition system will be presented as well as correlations of electron orbit, photon beam and vibrations of several spots will be shown.

INFRASTRUCTURE OF DATA ACQUISITION FOR VIBRATION

The DT8837 manufactured by Data Translation Inc. is employed as data acquisition tools for the accelerometers and photon intensity of beamlines distributed around the rings. The device supports functionality of bias current enable for ICP input. The equipped Ethernet interface is convenient for cabling and UDP trigger packet also provides sufficient synchronization mechanism for the distributed modules. Fig. 2 shows the infrastructure of the related system. All of the data from electron beam, photon beam, and vibrations could be synchronous acquired by software trigger within 100 msec. As Fig. 2 shown, besides the 10 Hz data from IOC/ILC could be acquired in real-time and archived, the fast transient motion could be also observed in adjustable higher time resolution and sampling rate up to 10 kHz.

Figure 1: TPS construction site in Sep 2010.

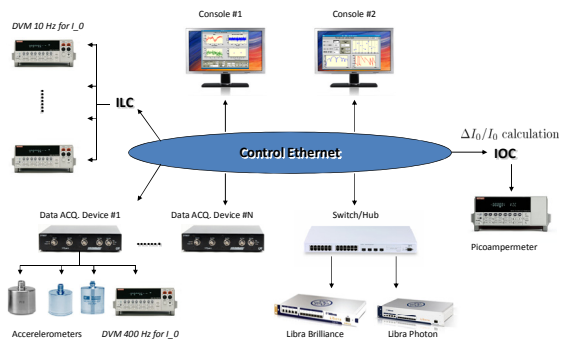
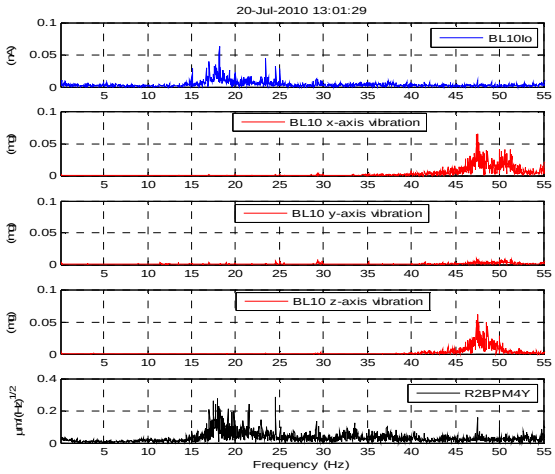


Figure 2: Infrastructure of data acquisition for vibration and the other related subsystem.

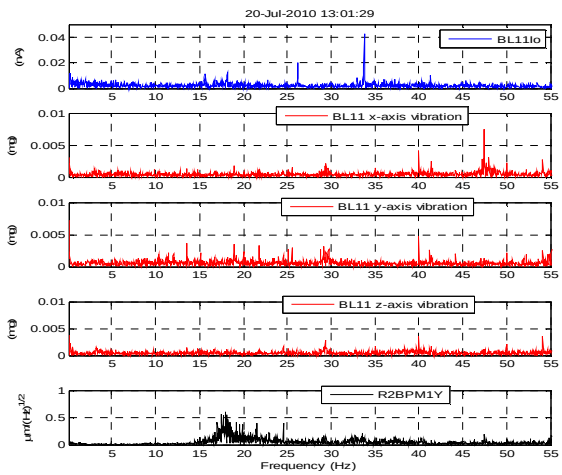
STATUS OF NORMAL OPERATION

Fig. 2 shows the normal status of the beam stability in quiet: the stability of beamline intensity (Δ Io/Io) is usually under 0.1%, the spectrum amplitude of electron beam stability is also less than 0.5 um below 50 Hz. The overall RMS stability of electron beam can achieve submicron level from DC to 50 Hz in normal operation [1][2]. The mechanic design of BL11 looks better than BL10's where the vibrations of three-axis at BL11 are all

less than 0.01 mg and are less 0.1 mg at BL10 as Fig. 3. It is required further studied to seek for causes of these differences. It is clear from Fig. 3 that the spectrums of two electron BPM are very similar while they are not consistent with the spectrum of photon intensity Io between BL10 and BL11. Furthermore, even these two Io cannot agree with each other. The vibration characteristics of two beamlines are not consistent so much. The fact that 47.5Hz noise is observed in both of the electron beam and accelerometers but is invisible in photon beam. It seems very necessary with more sensors for detection and analysis.



(a)

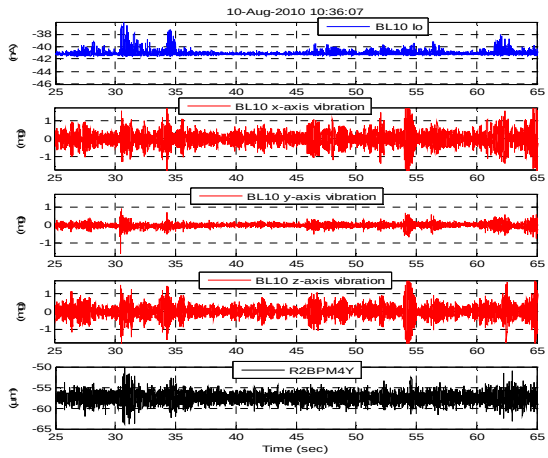


(b)

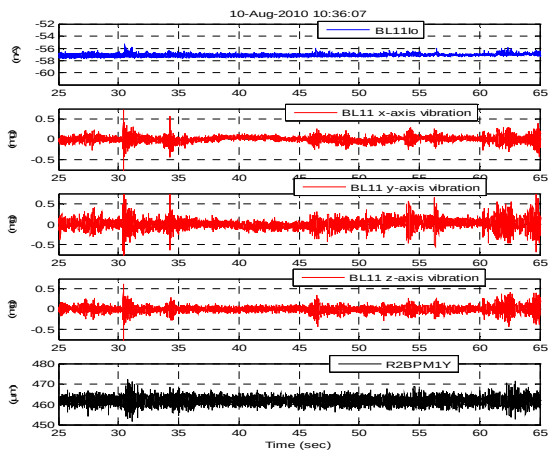
Figure 3: (a) Spectrum of BL 10 Io, three-axis vibration and electron BPM: R2BPM4Y. (a) Spectrum of BL 11 Io, three-axis vibration and electron BPM: R2BPM1Y.

LARGE VIBRATION CONDITION

The inconsistence of the above section is also presented when large vibration occurs. Although the scales of the instabilities of photon intensity and vibration became larger when excavators or pile drivers were operated, but the characteristic of the behavior is still quite differed. Fig. 4 shows one of the examples. It can be observed that in the time domain, the transient motions (spikes) occurred simultaneously but the spectrums of these signals aren't so correlated much as Fig. 5.

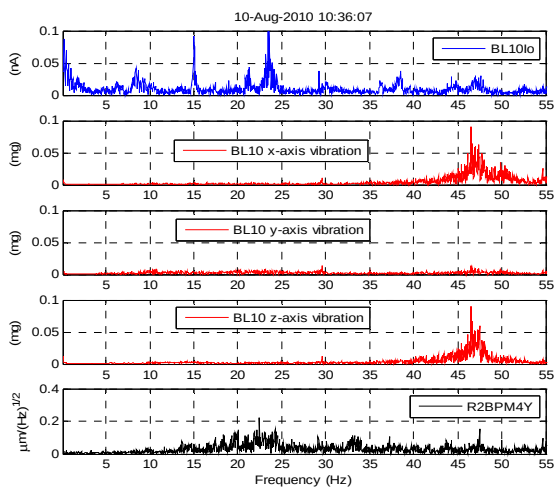


(a)

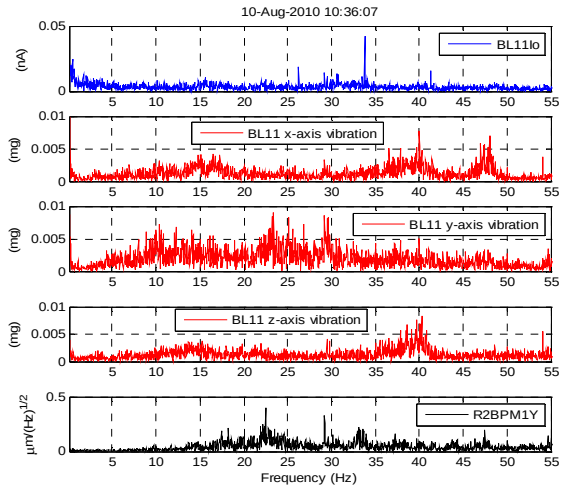


(b)

Figure 4: (a) Time series of BL 10 Io and three-axis vibration and electron BPM R2BPM4Y. (a) Time series of BL 11 Io and three-axis vibration and electron BPM R2BPM1Y when large vibration occurs.



(a)

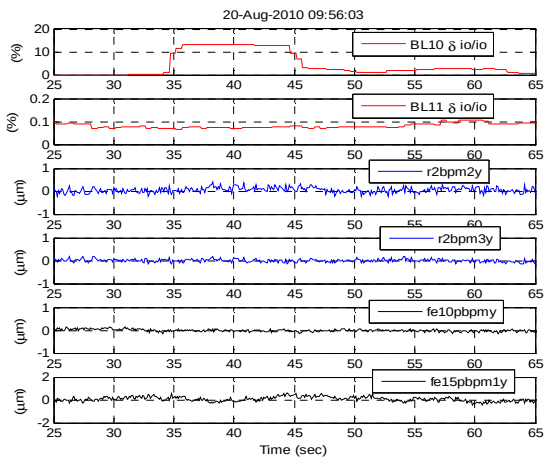


(b)

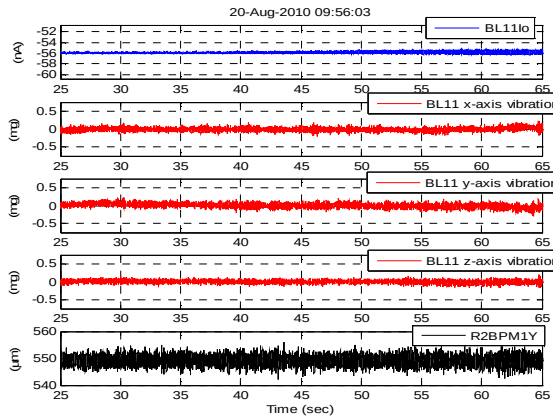
Figure 5: (a) Spectrum of BL 10 Io and three-axis vibration and electron BPM R2BPM4Y. (a) Spectrum of BL 11 Io and three-axis vibration and electron BPM R2BPM1Y when large vibration occurs.

⊿Io/Io change study

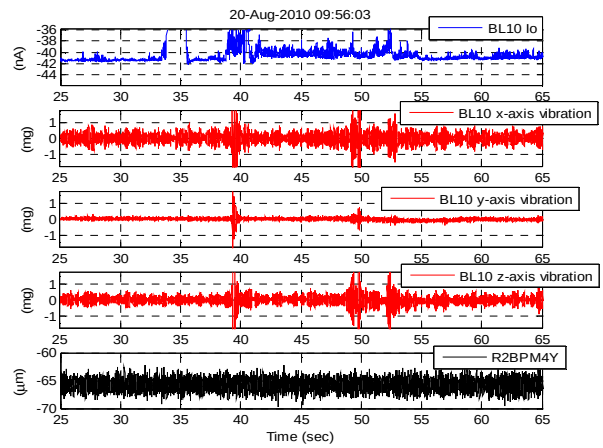
As Fig. 6 (a) shown, the stability indicators ⊿Io/Io of one beamline (in this examples, BL10) became worsen sometimes. In the meanwhile, the other (BL11) still remained normal and the electron beam orbit was also steady. We check the vibrations of these two beamlines at that moment as Fig. 4 (a) & (b). Transparently, an individual vibration event nearby this beamline caused the BL10 quake. The vibration was local not global. If not all of the indicators ⊿ Io/Io become worsen simultaneously, the indicators are meaningless. However, even if the global vibration result in instabilities, the change of photon beam motion is not majorly from electron beam but itself vibration contributed more. In fact, the electron beam is more stable than photon beam when large vibration occurs. The stability of photon beam deteriorated over twice while the electron beam almost not changed comparing Fig. 3 and Fig. 5.



(a)



(b)



(c)

Figure 6: (a) 10Hz data of ⊿ Io/Io and electron and photon BPM (b) Time series of BL 10 Io and three-axis vibration and electron R2BPM4Y. (c) Time series of BL 11 Io and three-axis vibration and BPM R2BPM1Y when large vibration occurs.

SUMMARY

The installation of the accelerometers and its data acquisition are presented. The vibration acquisition system provides information about ground vibration so that it could be correlated with electron and photon beam. It helps to clarify some unclear events and contradictions in the TLS operation. For examples, the inconsistency of ⊿Io/Io between different beamlines was possibly resulted from local ground motion. The characteristic of the different girders quite differed. The firmness of storage ring girder is better than beamlines and the electron beam are more immune from vibration than photon beam.

REFERENCES

[1] C. H. Kuo, et al., “Fast Orbit Feedback System Upgrade in the TLS”, Proceedings of ICALEPCS 2007.

[2] P. C. Chiu, et al., “Orbit Stability Observation of the Taiwan Light Source” , 2009 OCPA, January, 2009

COMPUTATIONAL STRATEGIES IN OPTIMIZING A REAL-TIME GRAD-SHAFRANOV PDE SOLVER USING HIGH-LEVEL GRAPHICAL PROGRAMMING AND COTS TECHNOLOGY

L. Giannone*, R. Fischer, K. Lackner and ASDEX Upgrade Team
Max-Planck Institut für Plasmaphysik, EURATOM-IPP Association, D-85748 Garching, Germany
P.J. McCarthy
Department of Physics, University College Cork, Cork, Ireland
Q. Ruan, A. Veeramani, M. Cerna, J. Nagle, M. Ravindran, D. Schmidt, A. Vrancic, L. Wenzel
National Instruments, Austin, TX 78759-3504, Texas, USA

Abstract

Big physics control experiments require enormous computational power to solve large problems with demanding real-time constraints. Sensors are acquired in real-time to feed mathematical routines, which then generate control outputs to real-world processes. For tokamak control, a non-linear PDE needs to be solved in real-time with a cycle time of less than 1 ms.

We report on an alternative approach based on LabVIEW that solves the critical plasma shape and position control problems in tokamaks. Input signals from magnetic probes and flux loops are the constraints for a non-linear Grad-Shafranov PDE solver to calculate the magnetic equilibrium. An architecture based on off-the-shelf multi-core hardware and graphical software is described with an emphasis on seamless deployment from development system to real-time target. A number of mathematical challenges were addressed and several generally applicable numerical and mathematical strategies were developed to achieve the timing goals. Several benchmarks illustrate what can be achieved with such an approach.

INTRODUCTION

The magnetic equilibrium for a tokamak is described by the Grad-Shafranov equation :

$$R\frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial \psi}{\partial R}\right)+\frac{\partial^2 \psi}{\partial Z^2}=-\mu_0 R\mathbf{j}(R,Z), \qquad (1)$$

where ψ is the poloidal flux function, **j** is the current density, R is the radial component and Z is the axial component (see figure 1). This problem is commonly solved by a cyclic reduction algorithm [1, 2, 3]. A magnetic equilibrium for discharges with plasma current is reconstructed on a 33 x 65 grid using 40 magnetic probes and 18 flux loop difference signals. The right hand side current density term is calculated by a weighted least squares fit to the measurements which yields coefficients for the basis current density profiles [2, 3, 4]. Three basis current density profiles were chosen in the first round of development and found to adequately fit the experimental magnetic probe and flux loop measurements [5]. The currents from the poloidal field coils are also needed to compute the value of ψ on the spatial grid.

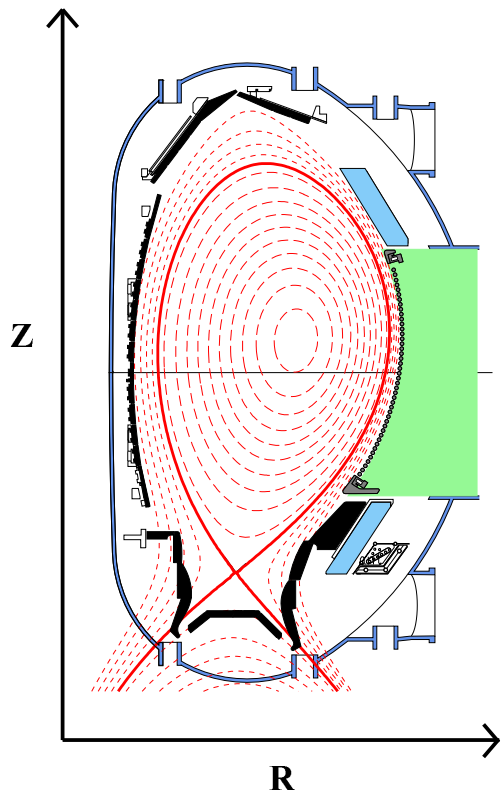


Figure 1: The cross section of the ASDEX Upgrade tokamak showing the flux surfaces of the magnetic equilibrium (red dotted lines) and plasma separatrix (red solid line).

REAL TIME GRAD-SHAFRANOV SOLVER

We report on a new spectral-based algorithm to solve the Grad-Shafranov equation in an unbounded domain. The new algorithm adapts a method commonly used to solve the Poisson equation in cylindrical coordinates. The use of discrete sine transforms (DST) along the Z-axis and a

* Louis.Giannone@ipp.mpg.de

tridiagonal solver [6, 7] is an alternative to the cyclic reduction algorithm to solve the Grad-Shafranov equation for poloidal flux, ψ.

Spectral Method

A uniform mesh with constant spacing dR and dZ in the R and Z directions is assumed. The grid points are labeled from 0 to $NZ-1$ and 0 to $NR-1$, where NZ is the number of grid points in the Z direction, and NR is the number of points in the R direction. The five point difference equation with index i in the R direction and index j in the Z direction can be written as :

$$\frac{\psi_{i+1,j}-2\psi_{i,j}+\psi_{i-1,j}}{dR^2}-\frac{1}{R_i}\frac{\psi_{i+1,j}-\psi_{i-1,j}}{2dR} + \frac{\psi_{i,j+1}-2\psi_{i,j}+\psi_{i,j-1}}{dZ^2}=-\mu_o R_i \mathbf{j}_{i,j} \quad (2)$$

Introducing the discrete sine transform of ψ and $\mathbf{j}$:

$$\phi_{i,k}=\sum_{j=1}^{NZ-2}\psi_{i,j}sin\left(\frac{\pi jk}{NZ-1}\right) \quad (3)$$

$$J_{i,k}=\sum_{j=1}^{NZ-2}\mathbf{j}_{i,j}sin\left(\frac{\pi jk}{NZ-1}\right) \quad (4)$$

leads to the tridiagonal matrix equations :

$$\beta_i\phi_{i+1,k}-\alpha_k\phi_{i,k}+\gamma_i\phi_{i-1,k}=-\mu_0 R_i dR^2 J_{i,k} \quad (5)$$

where $\alpha_k = 2+4S^2sin^2\left(\frac{\pi k}{2(NZ-1)}\right)$, $\beta_i = 1-dR/(2R_i)$, $\gamma_i = 1+dR/(2R_i)$ and $S=dR/dZ$.

Tridiagonal Solver

The tridiagonal matrix equation is solved with a tridiagonal solver using an LU decomposition algorithm. The LU decomposition generates two bidiagonal matrices subsequently used in the iterative procedure to solve the tridiagonal equations. By using LU decomposition, operations are reduced by a factor of 2 compared to the direct solver algorithm [8].

Unbounded Domain

The solver for the Grad-Shafranov equation in an unbounded domain is composed of two fast solver steps [1]. The new algorithm reduces the computing time dramatically by utilizing a spectral method at each step.

The first step of the solver uses zero as the condition for all grid boundaries with a right hand side current distribution on the flux surfaces from the previous iteration given by the weighted least squares fit to the magnetic probe and flux loop measurements. In this step, it is only necessary to compute ψ at points neighboring the grid boundary and a reduced inverse DST can be performed to calculate these values. The columns of ψ inside the boundary edge are :

$$\psi_{i,k}=\frac{2}{NZ-1}\sum_{j=1}^{NZ-2}\phi_{i,j}sin\left(\frac{\pi jk}{NZ-1}\right) \quad (6)$$

where $i=1$ and $NR-2$, and the rows inside the boundary edge can be calculated in a similar fashion with $k=1$ and $NZ-2$. All these four edges can be computed using matrix-vector multiplication. This avoids the unnecessary computations performed by a traditional inverse DST operation applied to the entire grid. The gradients in ψ normal to the grid boundary, $(\partial\psi/\partial n)_{boundary}$, are the inputs required for the next solver step. These are the shielding currents that are necessary to force the zero boundary condition of the first solver step. They are used to calculate the Green's functions for ψ generated by a current hoop of radius, a, carrying current, I, for each grid point with radial coordinate, R, and a vertical distance, Z, on the boundary [1, 9, 10] :

$$\psi=\mu_o I\sqrt{(a+R)^2+Z^2)}((1-k^2/2)K(k^2)-E(k^2)) \quad (7)$$

where $k^2=4aR/((a+R)^2+Z^2))$, $K(k^2)$ is the complete elliptic integral of the first kind and $E(k^2)$ is the complete elliptic integral of the second kind [11, 12]. The actual calculation of the resulting ψ on the boundary is performed as a matrix multiplication with pre-calculated coefficients times the vector of shielding currents.

The second step of the solver is carried out with boundary conditions from the first solver step but without current source terms on the right hand side of the Grad-Shafranov equation. Because only the first and last elements are nonzero, it is possible to use an optimized DST to reduce the computation effort. The faster DST is carried out by the BLAS function *dger* producing :

$$\begin{aligned} D_{ij} &= -\frac{\psi_{i,1}sin\left(\frac{\pi j}{NZ-1}\right)+\psi_{i,NZ-2}sin\left(\frac{\pi j(NZ-2)}{NZ-1}\right)}{dZ^2} \\ &= -\frac{\psi_{i,1}-(-1)^j\psi_{i,NZ-2}}{dZ^2}sin\left(\frac{\pi j}{NZ-1}\right) \end{aligned} \quad (8)$$

The DST of the boundary conditions at the inner and outer radial positions are added to the first and last columns. The tridiagonal solver is applied to this result and is added to the result from the first solver step. The solution of the Grad-Shafranov equation is then calculated by an inverse DST.

Under equivalent boundary conditions, an implementation based on the cyclic reduction algorithm computes all elements on the grid in both solver steps. The Grad-Shafranov solver algorithm described here achieves a significant performance improvement in comparison to cyclic reduction by employing two optimized DST implementations. The first implementation exploits the ability to avoid unnecessary calculations. The second implementation exploits the fact that the right hand side term is zero except at the boundary to greatly reduce the number of operations.

The ψ generated by the external poloidal field coils and passive stabilizing loop on the grid is also realized as a matrix-vector multiplication using factors calculated with Equation 7. The poloidal field coils and passive stabilizing loop are simulated as a finite number of filaments, with each filament carrying an applicable number of turns. Vacuum field shots with current pulses successively in each of the poloidal field coils are carried out to ensure that the best possible estimates of the magnetic probe and flux loop positions and calibration factors of the integrators are used to reconstruct the tokamak magnetic equilibrium with plasma current [5].

BENCHMARKS

A Dell T5500 with two PCI-e x16 slots wired as x8 (half length), two PCI-e x16 Gen 2 graphics slots up to 150 watts each, a PCI-X 64bit/100MHz slot with support for 3.3V or universal cards (half length) and a PCI 32bit/33MHz 5V slot (half length in desktop orientation) has been delivered with LabVIEW RT 2009 installed. A dual port Gigabit Ethernet card, a x4 PCIe VMIC 5565 PIORC reflective memory card and a NI PCIe 8362 interface card for connection of 2 PXI 1045 chassis for data acquisition of up to 256 channels were installed. Floating point benchmarks indicate a factor of up to 2.7 increase in performance in comparison to the current dual quad-core 3 GHz Xeon 5365 computer currently used for data acquisition and real-time calculations of magnetic equilibrium using only function parameterization [13]. The reflective memory card transmits the 33x65 poloidal flux matrix value to the control system with less than 1 ms delay. A third party PCI card delivers 64 bit time stamps using a 100 MHz clock and generates the 10 MHz TTL pulses for clock synchronization of the data acquisition boards in a number of data acquisition systems.

The following cycle time benchmarks were achieved for the real time Grad-Shafranov solver (GS) :

Table 1: Benchmarks for a single iteration of the real-time Grad-Shafranov solver (GS) using 8 CPU cores and LabVIEW RT 2009.

Platform	**GS (ms)**
Xeon X5365 @ 3.0 GHz	1.13
Xeon X5677 @ 3.46 GHz	0.63

The achieved cycle time for the Grad-Shafranov solver is therefore satisfactory for the real-time processing requirements of neoclassical tearing mode stabilization experiments where the cycle time of the discharge control system is 1.3 ms [14]. It should be noted that these benchmarks are for a single cycle iteration for the PDE solution. A detailed comparison of real-time magnetic equilibrium reconstruction with well converged solutions from offline calculations show that the small differences that are found for relatively steady state conditions are not relevant for practical discharge control [2].

CONCLUSION

A real-time Grad-Shafranov solver based on a discrete sine transformation of the difference equation rather than cyclic reduction has been realized. The resulting tridiagonal equations are solved with a specially developed subroutine based on LU factorization. This tridiagonal solver reduces the number of operations with respect to the iterative direct solver by pre-calculating the reciprocal of the diagonal elements. A reduced inverse DST is required in the first solver step as only the relevant terms for those neighbors of the grid boundary need be calculated. A simplified DST can used for the second solver step where only the first and last elements are non-zero. In this way the full inverse DST of the first solver step is omitted and the DST of the second solver step without current source terms can be calculated with a smaller number of operations. The real-time Grad-Shafranov solver cycle time of 0.63 ms on the delivered Dell T5500 platform satisfies the ASDEX Upgrade real-time processing requirements.

REFERENCES

[1] K.Lackner, Comp. Phys. Comm., **12**, 33 (1976)

[2] J.R.Ferron, M.L.Walker, L.L.Lao et. al, Nuc. Fusion **36**, 1055 (1998)

[3] P.J. McCarthy, Physics of Plasmas, **6**, 3554 (1999)

[4] W.Zwingmann et. al, Plasma Phys. Control. Fusion, **43**, 1441 (2001)

[5] L.Giannone, R.Fischer, J.C.Fuchs et. al, `http://ocs.ciemat.es/EPS2010PAP/pdf/P4.122.pdf`

[6] R.W.Hockney and J.W.Eastwood, "Computer simulation using particles", p208, Taylor and Francis (1988)

[7] M.H.Hughes, Comp. Phys. Comm., **2**, 157 (1971)

[8] W.H.Press, S.A.Teukolsky, W.T.Vetterling and B.P.Flannery, "Numerical Recipes in C", p50, Cambridge University Press, (1999)

[9] J.D.Jackson, "Classical Electrodynamics", p142, Wiley, (1999)

[10] J.Simpson, J.Lane, C.Immer and R. Youngquist, Simple analytic expressions for the magnetic field of a circular current loop `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20010038492001057024.pdf`

[11] M.Abramowitz and I.A.Stegun, "Handbook of Mathematical Functions", p591, National Bureau of Standards (1972)

[12] M.Abramowitz and I.A.Stegun, "Handbook of Mathematical Functions", p588, National Bureau of Standards (1972)

[13] L.Giannone, W.Schneider, P.J.McCarthy et. al, Fusion Eng. Des. (2008), `http://dx.doi.org/10.1016/j.fusengdes.2008.12.059`

[14] W.Treutterer, L.Giannone, K.Lüddecke et. al, Fusion Eng. Des. (2008) `http://dx.doi.org/10.1016/j.fusengdes.2008.12.026`

ESS CONTROLS STRATEGY AND THE CONTROL BOX CONCEPT*

T. Satogata#, Jefferson Lab, Newport News, Virginia, U.S.A.
I. Verstovsek, K. Zagar, and J. Bobnar, Cosylab, Ljubljana, Slovenia
S. Peggs and C.G. Trahern, ESS, Lund, Sweden

Abstract

The European Spallation Source (ESS) will be constructed by a number of geographically dispersed partner institutions in an international collaboration [1]. This increases organizational risk, as control system integration will be performed by a large number of quasi-independent teams. Significant effort will be put into standardization of hardware, software, and development procedures early in the project. The ESS will use EPICS, and will build on the positive distributed development experiences of SNS [2] and ITER [3-5]. The basic unit of standardization is called the Control Box. This consists of one or more input/output controller (IOC) computers, zero or more I/O modules, PLC subsystems, and intelligent special-purpose controllers, and includes software and integrated development environment support. We present the challenges faced by Control Box plans for ESS, and expected benefits.

INTRODUCTION

Lund was chosen as the ESS site in May 2009. The Design Update phase (Jan 2011 to Dec 2012) will be completed with delivery of a Technical Design Report (TDR). ESS will deliver proton beam through a ~420m superconducting linac, and is expected to begin delivering beam to users in 2019. ESS will eventually deliver a nominal average proton current of ~50 mA at ~2.5 GeV in ~2 ms long pulses with a repetition rate of ~20 Hz to a single neutron target station, for a nominal average beam power of 5 MW.

There are several base assumptions for ESS control system planning:

- ESS will use the EPICS control system.
- ESS will use the Linux operating system in the controls service tier.
- ESS will use the Oracle relational database system as a project-wide RDBMS.

After approval of the CDR in late 2012, the ESS project will proceed with R&D and construction, installation, and commissioning. ESS partner institutions doing development and R&D work over many geographical locations will be supplied with Control Boxes and given tools to enforce standards for common data management issues such as naming conventions, source code control, and controls development environment.

THE CONTROL BOX CONCEPT

The SNS project faced similar distributed controls and integration development challenges [2]. Several later projects, particularly ESS and ITER, are following the SNS distributed collaborative accelerator construction model and also require early broad controls coordination.

The Control Box concept is similar to the Plant System Host (PSH) concept used in ITER controls development [3]. In ITER terminology, the Control Box philosophy is realized with the concepts PSH, mini-CODAC [4], and Plant System I&C (instrumentation and control). The main purposes of the Control Box are to:

- allow independent and yet standardized subsystem controls development,
- enforce consistency between subsystems (possibly including target and experimental stations),
- facilitate testing of new components (e.g. EPICS drivers),
- allow centralized acceptance testing of subsystems through the control system,
- validate technology decisions,
- reduce risks early to lower projection integration uncertainty and effort,
- force early documentation of standards,
- and minimize throw-away hardware and software development.

An example structure of an ESS Control Box is shown in Fig. 1. The ITER Plant System I&C document [5] discusses the different available approaches to Control Box design.

*Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177.
#satogata@jlab.org

CONTROL BOX COMPONENTS

A generic ESS Control Box will consist of several software and hardware components:

- One or more EPICS input/output controller (IOC) computers.
- Zero or more I/O modules (analog-digital converters and digitizers, digital-analog converters, serial interfaces, etc.) attached to the IOC computer's hardware bus.
- A real-time or non-real-time operating system, depending on the requirements on IOC processing.
- A subset of the ESS EPICS real-time database to maintain values of all process variables under responsibility of the IOC.
- EPICS device support, which implements drivers for communication with equipment.
- EPICS Channel Access, which allows the process variables on the Control Box to be accessed from other computers in the network, and can retrieve values of process variables from other IOCs.
- PLC subsystems for slow industrial controls (e.g., water cooling; HVAC, etc), connected to the IOC with one of several standard communication mechanisms, such as PROFINET or Modbus TCP/IP.
- Intelligent special-purpose controllers (e.g. LLRF controllers).

A standard set of supported PLCs will be established during the ESS Design Update, similar to SNS and ITER. Intelligent controller development will occur as part of R&D and construction, and controller drivers will be shared with the EPICS collaboration.

The ESS Control Box distribution will package an EPICS distribution, Linux distribution, middleware, development environment, and documentation. This approach is similar to existing approaches by the NSLS-II and ITER projects. We are investigating IRMIS for project-wide PV management. Application-

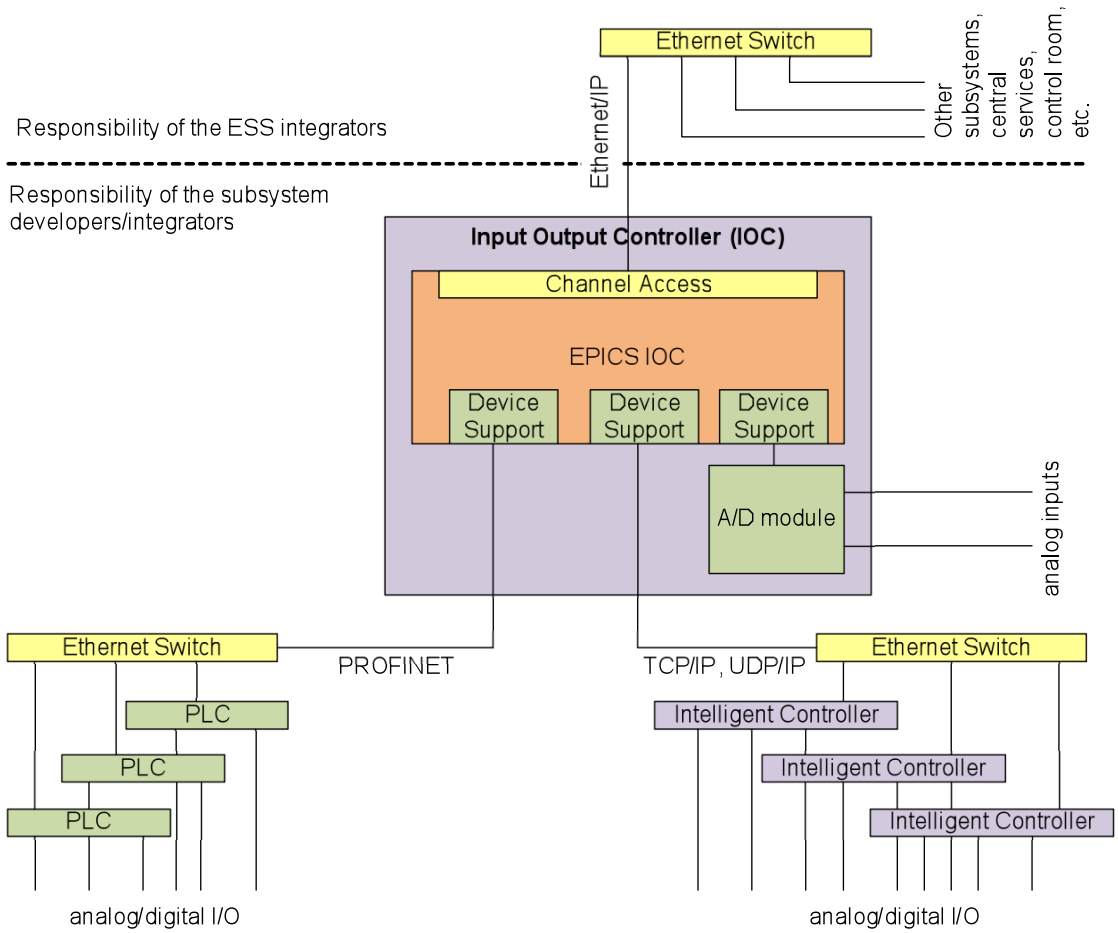


Figure 1: A schematic of Control Box components for ESS.

level development will use Control System Studio (CSS) [6] and XAL [7], and ESS and CosyLab are participating in a growing XAL collaboration [8]. A prototype of this Control Box package is a planned deliverable at the end of the ESS Design Update.

DATA MANAGEMENT

Data management among disparate R&D projects becomes another challenge that has control integration implications. In this area, one strength of EPICS (ease of adding and removing control points and IOCs) can also produce integration, maintenance, and diagnosis problems.

Central inventories will help manage this data and provide an infrastructure for consistency between distributed development and centralized machine design efforts. With limited resources, the ESS will focus on leveraging existing solutions such as the EPICS Channel Archiver [9] for historical values of process variables, CERN EDMS [10] for technical documentation and installation management, and IRMIS [11] for EPICS control inventory.

Project data integration during the design phase will be largely driven by the machine model in top-down design approach. A schematic of this approach is shown in Fig. 2. This also provides an infrastructure for naturally coordinating machine design through control system details such as lattice and control point names, such as in XAL.

DEVELOPMENT

Defining standards before R&D development may lower integration risk, but it raises technical risk. Controls development projections are quite uncertain nearly a decade from first delivered beam. Control Box development and support must therefore iterate through the R&D phase to react to changes in the technical landscape, incorporate new developments in EPICS, and distribute best use cases through the project. We plan to develop the ESS Control Box in annual cycles.

Early implementation costs are another challenge. The Controls Box concept requires enough maturity and management support at the outset that ESS development partners "buy in", and do not hide fragmentation beneath a layer of conformity. Early definition of naming standards is a priority of development, and agreement to adherence to these standards will be a requirement for ESS partners.

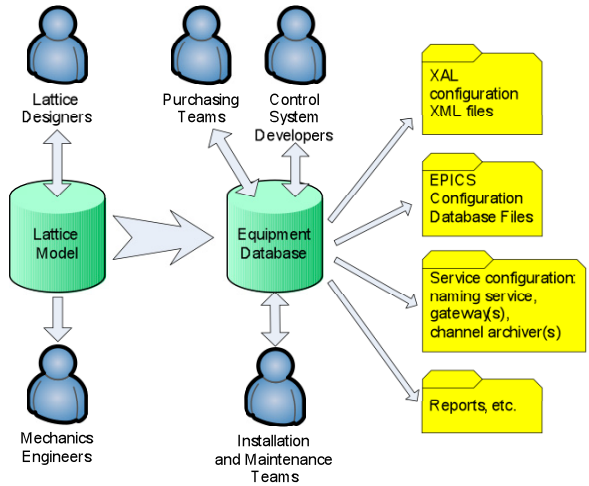


Figure 2: Flow of model data for top-down ESS control system design.

REFERENCES

[1] M. Eshraqi et al., "Conceptual Design of the ESS Linac", IPAC'10, Kyoto, Japan, June 2010, pp. 804-6.

[2] D.P. Gurd, "Management of a Large Distributed Control System Project", ICALEPCS'01, San Jose, California, Nov 2001, pp. 58-63.

[3] A. Wallander et al., "ITER Instrumentation and Control – Status and Plans", Fusion Engineering and Design **85**, July 2010, pp. 529-534.

[4] J.B. Lister et al., "The Status of the ITER CODAC Conceptual Design", ibid **83**, April 2008, pp. 164-9.

[5] A. Wallander, L. Scibile, Plant System I&C Architecture, http://tinyurl.com/2g4rhuy

[6] http://ics-web.sns.ornl.gov/css/

[7] http://www.ornl.gov/~t6p/Main/XAL.html

[8] J. Bobnar et al., "Applicability of XAL for ESS", these proceedings.

[9] http://ics-web.sns.ornl.gov/kasemir/archiver/

[10] http://edms.cern.ch/

[11] http://irmis.sourceforge.net/

CONTROL SYSTEMS FOR NEW LARGE EXPERIMENTS

J. Dedic, M. Plesko, R.Sabjan*, I. Verstovsek, K. Zagar, Cosylab, Ljubljana, Slovenia

Abstract

We discuss control systems of accelerators and similar projects that are presently still in design and early construction phases, such as FAIR [1], ESS [2], MedAustron [3], NSLS II [4], ITER [5], etc, and comparing them against the approaches of the last two decades and explain the new trends that are emerging:

- From the organizational perspective, control system architectures are established earlier in the project, allowing them to adapt to the machine physics requirements better as well as allow for modeling and simulations.
- In software, there is much less emphasis on custom codes than there was in the past. Instead, standard and off-the-shelf components and frameworks already used at existing accelerators are becoming the preferred choice, not only reducing risks, but also allowing for reuse and sharing.
- In hardware and networks for real-time control and data acquisition, there is a strong trend from custom electronics development to standard and off-the-shelf solutions. This in particular applies to systems like timing, machine protection, BPMs and LL RF. When custom solutions are needed, flexible hardware technologies (e.g., FPGA) are chosen to allow for future extensibility.

INTRODUCTION

Building a control system for a large experiment has always been a difficult task which required dedicated effort from a big group of people. And we have to thank controls groups in accelerator and the rest of big physics communities for their great achievements.

Control systems evolved in the recent decades, together with information technology, computer science and electrical engineering. In the starting days, little equipment, be it either software or hardware, was available off-the-shelf. A handful of physics labs with difficult requirements, for which solutions have never been implemented, were just not commercially interesting. This led to lots of custom work in the labs. From custom IO board development to advances in computer networking and developing whole software frameworks, nothing was taken for granted. Engineers were also scientists.

During the years, big number of experimental projects and the advance of computing allowed widespread standardization of components. Standard technologies are applied in every aspect of a modern control system, some systems can even be bought completely and some, which are only based on standard technology, but still require a lot of work before installed and commissioned. We shall look at some examples from the current experiments on which we collaborate.

At the end we shall try to summarize and find trends and consequences of progress. The main question is whether the everyday work of controls groups has changed and what does this mean for the main priorities that need to be set at the beginning of every project.

STANDARDIZATION IN LIGHT SOURCES

Plenty of light sources were built in the last decades and they have a lot common with respect to the control system. Control system packages (e.g. EPICS [6] or TANGO [7]) have matured through collaboration and can be easily deployed. They are supported on multiple standard hardware platforms (PC, VME, PXI etc.) and operating systems (Linux, Windows, Unix, Macintosh etc.). They provide solutions for most of your needs. Infrastructure applications like archiving, alarm handling or error logging are provided together with GUI builders and interfaces to many programming languages. Usually, even more than one implementation exists.

Increasing market has attracted industry as well. High performance electronics, made specifically for experiments' requirements is available off-the-shelf. Not only chips, but complete systems like digital BPM electronic [8] or timing systems [9] can be bought. Many equipment or subsystem vendors provide control system drivers with their products and they offer to implement them for the control system package of your choice.

Project leaders and funding agencies know this as well – control system budget has typically fallen from 10% to 5% of the machine's budget (not counting the building and beamlines). The challenge today is to implement a control system with state-of-the-art technology, but with a smaller budget and/or on a shorter time-frame, not sacrificing quality, of course. This prioritizes organizational aspects of the project which will be discussed in later sections.

PUSHING THE LIMITS OF CONTROL SYSTEM COMPONENTS

Other experiments (we have recently worked with ITER, FAIR, ESS and MedAustron) are still hiding more technical challenges and questions. Some examples are explained below.

Machine Protection System

One such example may be a complicated timing system or very flexible, but still safe machine protection system.

*rok.sabjan@cosylab.com

Reference (or just similar) implementations do not exist yet and these components are key to success of the whole project.

We have collected requirements from several projects and apart from the traditional role of the machine protection system (MPS) just statically reacting to digital inputs, new features are required. One such is a reconfigurable IO matrix, where responses to interlock inputs would be based on the current mode of the machine. This enables bypassing certain faults or threshold levels. Integration with the timing and control system is highly desired, allowing for quick reconfiguration of the system.

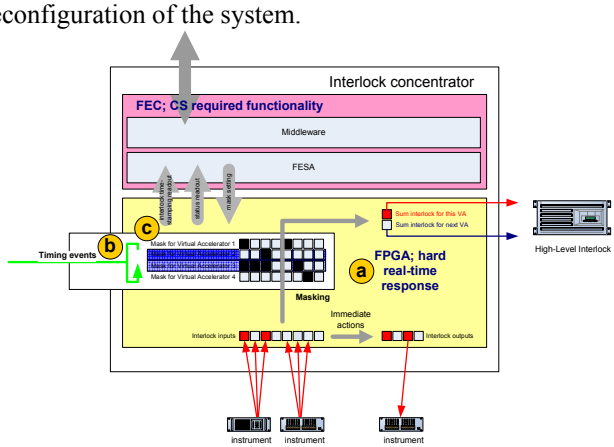


Figure 1: Possible implementation of a fast machine protection system

Integration with the timing system is important for the post-mortem analysis as well. Input signals can be accurately time-stamped and the proper timeline of a problematic event can be reconstructed even if there are a lot of interlocks firing.

Most of today's MPS implementations make use of PLC technology, which has a response rate in the range of several milliseconds. The new design allows response in the range of microseconds even with fibre lengths of over 1 km, making the speed of light the biggest constraint.

Such a standard solution does not yet exist, but the collaboration with a number of labs and their interest makes it worthwhile to start the development. It is our view that the solution to MPS can be a good mix of common general system with specific.

Hard Real-Time Feedback System

Another interesting control system component is a hard real-time feedback system, which brings distributed dimension to the real-time control. Implementations of this already exist and work well (e.g. fast orbit correction for storage rings).

However, current implementations are largely based on proprietary technologies like reflective memory (RM), or are implemented in-house using specialized solutions such as dedicated fiber network and custom hardware. We believe, in order to really standardize on open standards and to lower the cost and the risks for the future, (ten-) Gigabit Ethernet should also be considered.

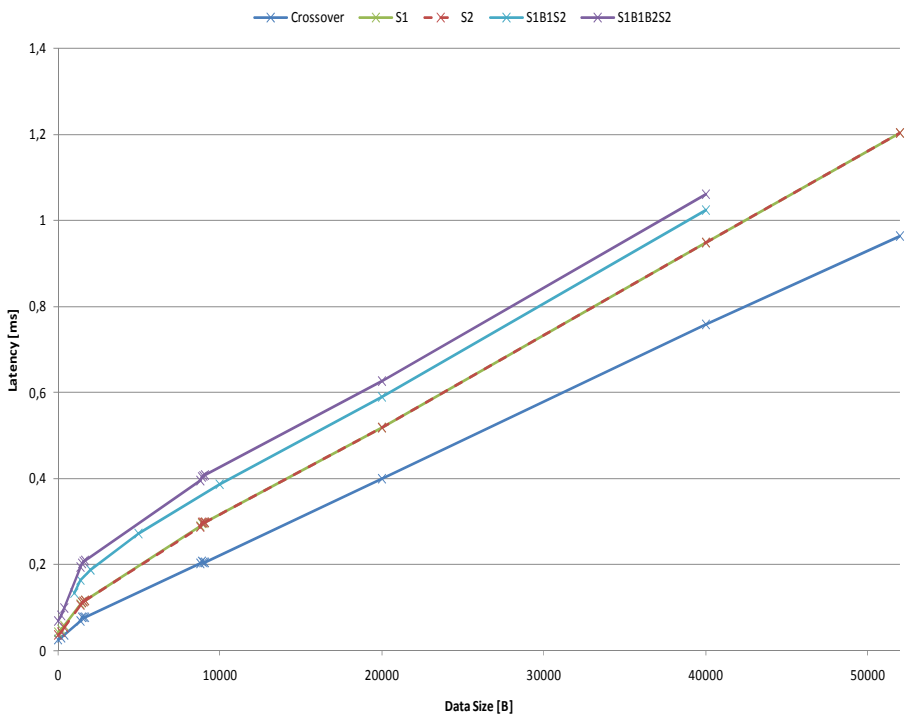


Figure 2: Measurement of latency using RTnet [10] and Gigabit Ethernet as a function of data size. Different lines represent different network topologies, from using a crossover cable only, to complicated topology, where four network switches are used.

We have measured the deterministic performance of Gigabit Ethernet as a task for ITER. We were interested in achieving 1 KHz feedback cycle (2 network hops per cycle) with very low jitter (less than 10 usec) with total traffic of 40kB per cycle. For this, we did not just look at standard UDP packets with multicasting over the network, but we also tested our setup with Xenomai [11] real-time Linux kernel and RT net, a real-time network stack implementation.

Our results showed (Figure 2) [12] that we can already achieve today a very good latency of 0.5ms for data rates that are typical for accelerators. Although we cannot use Gigabit Ethernet technology for ITER requirements today, we are very close. With 10-Gigabit Ethernet just years away, we are confident that Ethernet will be a very good choice for development efforts [13]. Commercially, no other technology can come close – consumer switches, network adapters and cables could be used. It seems very unlikely that this will change in the coming years.

Timing system

New complex machine require a timing system which is more complicated than just a simple event system that is usually used at light sources. New features like virtual accelerators, timing super-cycles (Figure 3) and event acknowledgements are introduced.

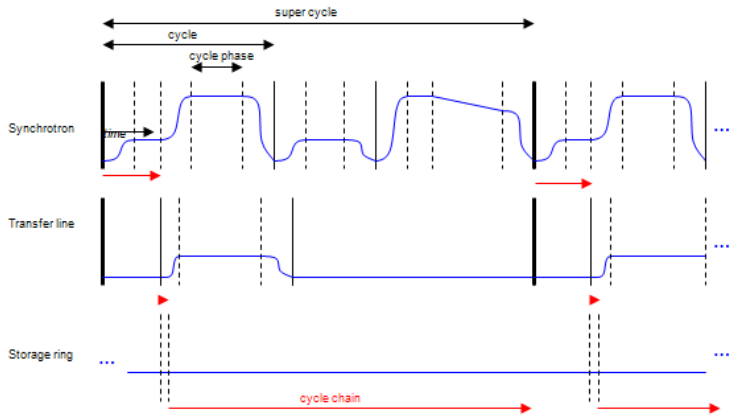


Figure 3: Example timing sequence for FAIR

The existing (off-the-shelf) timing solution like the one produced by Micro Research Finland, which is the most widespread among new machines, cannot provide all the needed functionality, but they can be used as the basic component, the transmission layer.

We see that despite having a commercially available standard solution a lot of customization work is necessary. You can purchase the transport layer, whereas the application layer is machine specific and needs to be implemented for every project individually.

COMPLEX COMPONENTS AND INTEGRATION

We have established that there are definitely trends towards standardization of control system components, which could mean that work is reducing for the controls team. But unfortunately, not everything is that simple.

Components are getting more complex and they require more time and effort to be integrated into your control system. Choices need to be made early in the project which is risky if not all aspects are considered.

Basic control system package

Traditionally the first choice is about the control system package itself (EPICS, TANGO, FESA, TINE, COACK, DOOCS, ACS etc). But this choice is not the most important one. In fact, we believe that people decide for a control system package in a similar way as when they are buying a car: we decide based on emotions and later we rationalize this discussion with architecture description and features. Luckily, most of control system packages are mature and modern technology will enable you to finish your project whatever your choice might be. That is why we recommend choosing the package that you like the most, either due to your personal experience, your people background or because a similar project already used it and those people can help you when you get in trouble.

Integrating other packages

Many facilities use more than just one control system, either they are dealing with a legacy system from a previous experiment, with a component developed by another group or buy machine components with existing commercial control system (e.g. NI LabView [14] or any other SCADA system). Typically, facility control (e.g. air condition) is already automated and needs to be integrated in the main system.

Usage of many different packages is to be avoided, if possible. For the remaining case, I believe that the main control system group must clearly define responsibilities and approve requirements for the interface, especially if other groups are involved. Documentation and

maintenance of the systems must be considered. Technical problems come second to interpersonal relationships in this case.

Another set of examples come from the machine physics world. There different packages are used, MatLab [15] and XAL [16] are the most popular recently. Issues here are all the interfaces to other control system components (process variables with all the attributes like alarms, relational database, event handling etc.).

Determining the level of integration is the most important issue to resolve. We need to realize that we are not just "pushing" the data from one system to another, but we must also think about configuration management and maintenance. For example, usually people have different views about which system will check values for alarm levels and how where these thresholds be defined.

In such cases it is usually best to adopt best practices developed and lessons learned by a previous similar experiment.

Distributed development and 'in-kind' projects

New large experiments, such as ITER, FAIR or ESS, are very costly and are often started as international projects with in-kind contributions. The extreme example of this is ITER, where more than 150 plant systems will be provided by the 7 collaborating countries together with the local control system that will be integrated into the main control system.

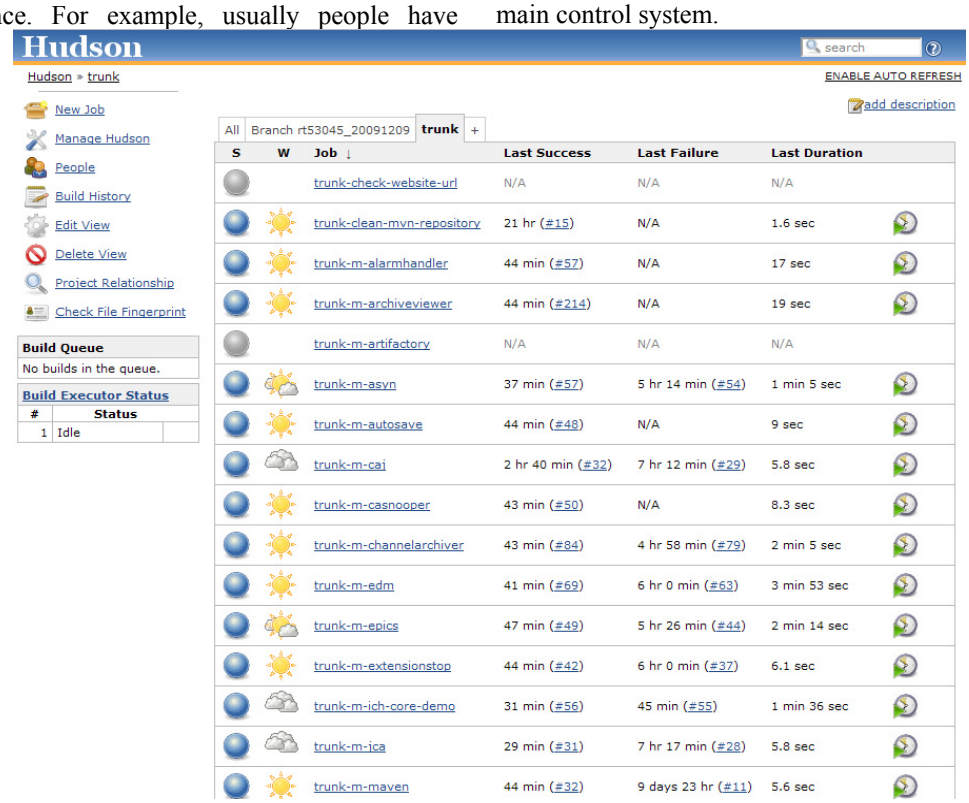


Figure 4: ITER Core System [17] is a software product helping to standardize and ease the development of the control system. To ensure the quality, the software is heavily covered with automatic unit tests which are run at every build. Continuous build system notifies the developers of build problems and test failings within a few minutes.

ITER is tackling this issue with very rigid standardization. Every year, the ITER controls group publishes the Plant Control Design Handbook (PCDH), which describes all the standards, and releases the Core System software (Figure 4), the set of all standard, ITER approved, community tools and software drivers.

The standardization does not stop with the main architecture, hardware platform and IO boards, operating system and software packages. Project life-cycle, naming convention and test plans are also specified.

In addition to this, the Core System software package is prepared. It is the practical aspect of the PCDH and will be used by all ITER collaborators, making it easier to develop the control system properly and easily.

We have recommended this approach to ESS as well and they have adopted the Control Box concept [18]. ESS will also be built by many partners, albeit not as many as ITER.

FOCUS ON DEVELOPMENT PROCESS

Building a complex system from more or less standard components is an engineering task (much more than a scientific experiment) with all the steps that are common to all engineering disciplines. In fact, control system development has an even more complicated cycle:

- Write specifications
- Architecture
- Design

- Prototyping – probably the only fun part
- Define test procedures
- Implementation (coding) – the only software part
- Writing documentation
- Testing (follow ISO procedures)
- Debugging
- Acceptance at customer

Projects are increasingly aware of the development processes. Especially, the international efforts recognize this and focus heavily on the following things. One such is the signal list. It is a golden list that represents the contract between different subsystems and different developers. This is very obvious and should be made in the initial stages, but many projects do not have it until very late in the project.

Signal list also requires a good naming convention, which is unique and still people-friendly. Different people need to access process variables in the control system and naming convention should help not hinder that.

Control groups are putting procedures in place that deal with changing signal list, hardware and software in manner that all interdependencies are taken care of and changes will be applied in all the appropriate places. This avoids project inconsistencies.

There are two more important procedures that are considered: logistics of installation and error handling (i.e. bug fixing). How one handles control system installation and testing needs to be defined well before integration time, even before any outsourcing contracts are written. It should define what are the necessary testing steps before integration starts, who is responsible for what part and what are the interfaces between different groups of people (control system people, device experts, subcontractors, electrical support team).

We all accept that some bugs are inevitable and sufficient time needs to be planned for testing and debugging. The procedure should also define how bugs are reported and how changes (fixes) are introduced and re-tested. Last by not least, good development practices minimize the number of bugs in the first place.

Big projects realize that man-power is a problem and it is difficult to cover the wide range of required competences. That is why they decide for outsourcing for a big part of control system, whereas they retain the overall system responsibility in-house.

CONCLUSIONS

Standardization is the key trend emerging with development of new and complex projects. Labs are not required to develop all parts of a control system themselves, but can rely on re-using development from other people or even buy off-the-shelf components and solutions. Technical risks are reducing.

Today, integration is the biggest aspect of a controls project. How will all the components fall into the main architecture, what will be the interfaces and how any of the requirements will be addressed, are the main questions. Integration starts with day one and is an every-day companion throughout the project.

Organizational risks in big and complex project with many partners are increasing. Focus needs to be shifted to stricter definition and implementation of development processes and rigorous standardization with clearly defined interfaces.

In short, control system development is becoming more and more an engineering discipline and less like a science.

REFERENCES

[1] FAIR; http://www.gsi.de/fair
[2] European Spallation Source; http://www.ess-scandinavia.eu
[3] MedAustron; http://www.ebgmedaustron.at
[4] NSLS-2, National Synchrotron Light Source 2; http://
[5] ITER, ; http://www.iter.org
[6] EPICS collaboration; http://www.aps.anl.gov/epics
[7] TANGO collaboration; http://www.tango-controls.org
[8] Instrumentation Technologies; http://www.i-tech.si.
[9] Micro Research Finland; http://www.mrf.fi
[10] RTnet: Hard Real-Time Networking for Real-Time Linux; http://www.rtnet.org
[11] Xenomai: Realtime Framework for Linux; http://www.xenomai.org
[12] K. Zagar et al, "Evaluation of High-Performance Network Technologies for ITER", 7th Technical Meeting on Control, Data Acquisition and Remote Participation for Fusion Research, Aix-en-Provence, June 2009
[13] K. Zagar, "Ethernet-based Real-time Networks for Distributed Cloosed-loop Control", PhD Thesis, to be published
[14] National Instruments; http://www.ni.com
[15] MatLab CA; http://ics-web.sns.ornl.gov/kasemir/mca
[16] XAL; http://www.ornl.gov/~t6p/Main/XAL.html
[17] K. Zagar et al, "ITER control system development environment", this conference proceedings.
[18] T. Satogata et al, "ESS Controls Strategy and Control Box Concept", this conference proceedings.

PRESENTATION ONLY

WHITERABBIT - A NOVEL, HIGH PRECISION TIMING SYSTEM

M. Kreider, R. Baer, T. Fleck, C. Prados (GSI, Darmstadt)
E. Garcia Cota, J. Serrano, T. Wlostowski (CERN, Geneva)

Abstract

The WhiteRabbit timing network is a deterministic field bus, based on synchronous GBit Ethernet and the Precision Time Protocol (PTP). The WR protocol was designed to provide precise timing and event distribution for high end real-time systems and was therefore chosen as the timing basis for the new GSI FAIR accelerator facility. With precise phase measurement to compensate for signal propagation delay, a timing accuracy down to sub-nanosecond range is feasible. To achieve necessary determinism and robustness (packet loss of 10^{-12}), an OSI layer two Forward Error Correction and Quality of Service protocol have been introduced to the concept. Special switches wield the WR protocol, while being transparent to normal Ethernet traffic. Switch hardware is currently under development at CERN and will be a mixed FPGA/CPU solution. Working prototype cards have been introduced at the 3rd WR Workshop at CERN in 2009, demonstrating phase measurement and PTP capabilities. The presentation will contain detail on technical concepts, current project status, as well as future areas of application will be part of the discussion.

INTRODUCTION

Purpose

WhiteRabbit was designed to provide very accurate clock synchronisation to a facility and control its machines with equal precision. Any event sent to a physical machine causes a certain action to be executed at a given absolute time.

The goal here is to the know the exact link delay to destination in advance, so each outgoing event can be sent out early enough to arrive on time.

In order to achieve that, certain unpredictable factors to the response time have to be addressed. One is packet loss due to data corruption on the physical medium, the other factor is collisions resulting from packet switching in the network.

NETWORK LAYOUT

WR utilises GigaBit Ethernet on fiber or copper links. Fiber links have an advantage here, because copper tranceivers and their channel encoding logic are more complex and often show a non-determinstic behavior. Optical links enable a higher measurement accuracy on link delay.

The topology of WR system may take any non-meshed form, since time synchronisation must be unidirectional. If the network is indeed meshed, a Spanning-Tree algorithm must be used to avoid loops in time distribution.

GSI/FAIR is planning to employ a Tree Topology with a GPS receiver as UTC timing reference at the source. Below come several layers of switches, fanning timing out to endpoints throughout the facility.

WR uses special switches and endpoints to wield its protocol. Current design of the WR switch has one uplink and sixteen downlinks, each has a second physical port for redundancy. GSI/FAIR is planning a system with roughly two thousand timing receivers

Making extensive use of commercially available ethernet basic components lowers costs for WR switches and endpoints. It will be possible to integrate non-White rabbit nodes into the network. WR is compatible with PTP devices and can time sync these nodes. However, PTP nodes can only be synchronized with reduced accuracy, since they lack the special hardware for high precision phase measurement.

General purpose ethernet nodes could also be connected to the network. While being compatible with basic functions, WhiteRabbit design does not support full Ethernet standard at the time.

TECHNOLOGY

Synchronous Ethernet - SyncE

SyncE describes the special case of IEEE 802.3 ethernet standard where the recovered RX clock from its master is used as its own TX clock, making the whole system synchronous. 8b/10b channel encoding is used to make RX clock recovery from the incoming RX data signal possible. This adjustment is done in hardware and is the basis for the PTP fine measurements.

Phase Measurement - Aliasing and DPLL

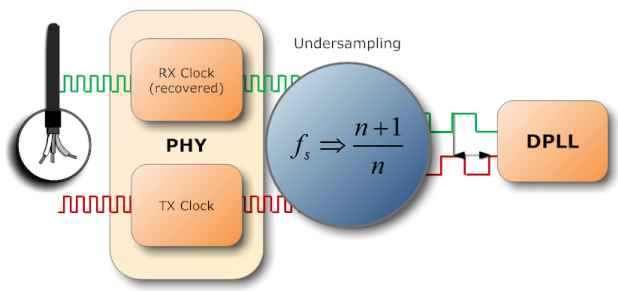


Figure 1: Aliasing and Phasemeasurement

After SyncE has adjusted the PTP clients frequency to the masters, the PTP can now measure the time difference

and lag between nodes. With WR, this is aided by hardware doing the precise measurement on the clocks phase difference, bringing timing accuracy from 8ns to a theoretical value of 32ps. In order to get highest precision, the clocks frequency would have to be in optimal range of the PLL.

The endpoint achieves this by undersampling both clocks with a frequency very close to their own. Assuming mid term stability of the oscillator, this produces low alias frequencies which lie in the optimum measuring range and still possess the proportional phase shift of the original while jitter is greatly reduced .

Time synchronisation - PTP

PTP addresses the basic problem of clock synchronisation when message lag and local time difference are unknown. A handshake between master and client is initialised, all incoming and outgoing messages are timestamped. After two messages and four time values, it is possible for the master to calculate link delay and difference of localtime. The master then communicates the correctional value down to the client which adjusts its own clock.

WhiteRabbit uses an extended version of the IEEE 1588 Precision Time Protocol. Here, synchronisation direction is fixed, hardware phase measurement increases accuracy and assymetry in link delay is taken into account.

This assymmetry is a result of chromatic dispersion coming from wavelength multiplexing in the medium. A single fiber is used for both RX and TX, employing two different wavelengths to differ between incoming and outgoing messages. Light propagation in a an optical fiber is a function of its wavelength, so RX and TX will differ in propagation delay. [1]
Choosing fiber type with a nearly equal dampening for each wavelength helps balancing signal strength and therefore maximum range.

Time is adjusted sequentially down the layers of switches and nodes. Further consideration for the link delay model are slowly changing characteristics of the physical medium, caused by temperature, moisture and aging effects.

Encoding - Forward Error Correction

The goal for WR is an event loss of 10^{-12}. Normal TCP protocol for example handles the problem of data loss by re-requesting the damaged packet. In WR, there is no time for this backup mechanism. Forward Error Correction algorithms are a class of encoding that can introduce enough redundancy to the data that chances of a packet being irrecontructably lost are minimimal. An event stream always consists of several packets, and the packet header is additionally secured with a CRC check. Individual packets themselves may be lost or corrupted, the event must reach its destination nevertheless. A detailed description and analysis of suitable algorithms and effectiveness is available here [5].

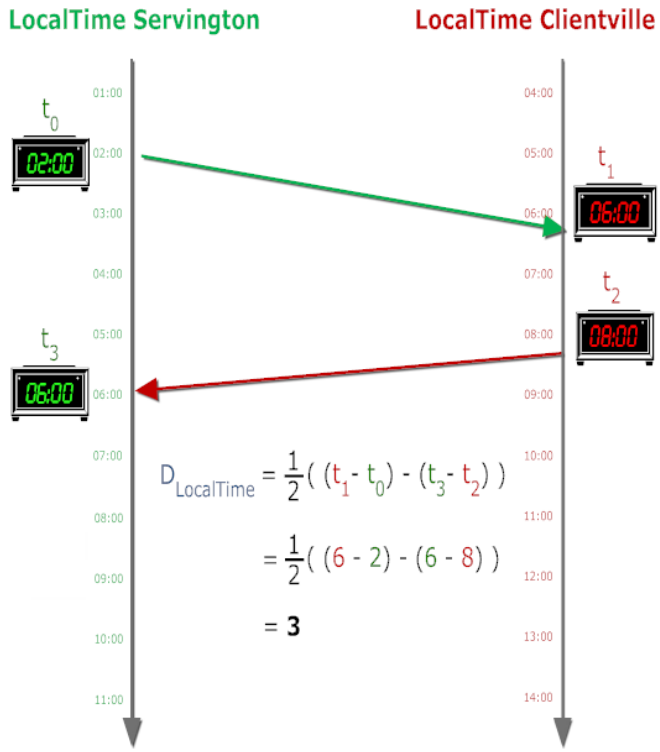


Figure 2: Simplified PTP Delay Calculations

Packet Switching - QoS

In order to guarantee absolute maximum lag time, it is necessary to prefer time critical packets to others. When a switch has more than one output packet for a port at a time, a second arriving packet must be treated differently depending on priority. Standard Priority packets , SP, can be queued if buffer space allows, else they are dropped. SP packets are not time critical and therefore re-requesting a dropped packet is possible.

Any arrival of an HP packet will cause a currently sent SP packet to be fragmented and resumed when HP traffic is over. This makes the 64 cycles maximum delay when crossing a switch possible.

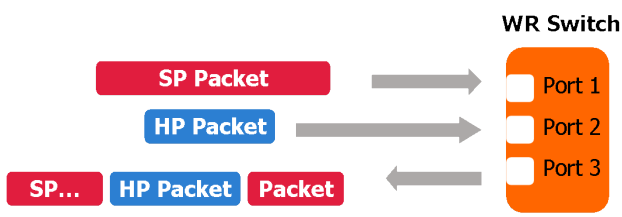


Figure 3: HP packet preempting SP packet

OPEN HARDWARE PROJECT

When thinking about the implementation of WhiteRabbit, care has been taken from the beginning not to use commercial components that come with royalty fees. At the same time, WR needed protection against possible lawsuits for suggested patent infringements or similar.

WhiteRabbit Hardware and Software is completely open and documentation and sources are available at `http://www.ohwr.org/`

CONCLUSION

Working Point-To-Point time synchronisation has first been shown at the WR workshop in 2009. Since then, switch hardware was under continous development and a first WR switch prototype with switching capabilities will be ready by the end of 2010.

Running side by side with hardware development, WR protocol specs were expanded and improved. Timing Receiver boards are also currently under development and will be made in various form factors. First planned are PCIe and VME boards to accomodate a timing receiver in many existing systems, first prototypes are expected early 2011.

WhiteRabbit is a timing system for the future. GSI is planning to control all its current and new FAIR machines over WhiteRabbit, with the only exception of the HF generators.

A fully deployed system at the FAIR facility is to be expected by 2016.

REFERENCES

[1] P.P.M. Jansweijer, H.Z. Peek, “Measuring propagation delay over a 1.25 Gbps bidirectional data link”, ETR 2010-01, Amsterdam, Netherlands, May 2010

[2] P. Moreira et al., “White Rabbit: Sub-Nanosecond Timing Distribution over Ethernet”, ISPCS 2009, Brescia, Italy, Oct 2009

[3] J. Serrano et al., “THE WHITE RABBIT PROJECT”, TUC4 ICALEPS2009, Kobe, Japan, Oct 2009

[4] WR Switch Specifications `http://www.ohwr.org/`

[5] C. Prados Boda, T. Fleck, “ FEC in Deterministic Control Systems over Gigabit Ethernet”, THPL011 PCaPAC2010, Saskatoon, Canada, Oct 2010

FLASH DAQ DATA MANAGEMENT AND ACCESS TOOLS

* V.Rybnikov, V.Kocharyan, K.Rehlich, E.Sombrowski, T.Wilksen

Abstract

The Free Electron Laser in Hamburg (FLASH)[1] at DESY is a user facility for the photon science community. It produces laser light of short wavelengths from the extreme ultraviolet down to soft X-rays. To study, monitor and document the machine performance and parameters and also to collect the results of the experiment measurements, a fast data acquisition (DAQ) system is being used. Having above 1000 linear accelerator diagnostics channels collected by the DAQ currently results in a data rate of ~100 Mb/s. The large amount of data requires corresponding data storage and management to enable efficient data retrieval. This paper will focus on the data paths, storage and bookkeeping. A number of tools provided for the users to work with DAQ data will be described. The current status of the achieved performance in the data storage and retrieval will be covered as well.

INTRODUCTION

The FLASH DAQ [2] system was launched in summer 2004. Its main tasks are: collecting LINAC beam relevant data in real time, providing the data to feed-back and monitoring tools as well as storing it for an offline analysis. The DAQ system is also used by FLASH user experiments to store their data together with information coming from LINAC. This allows easy correlations between the experiment measurements and the LINAC state. A set of tools is provided for data visualization and analysis.

DATAFLOW

The dataflow in the FLASH DAQ and all involved components are shown in Fig. 1. There are two types of data collected by the DAQ. Fast data include channels with beam related information (beam position monitors, etc.) and currently collected with the shot repetition rate of 10 Hz. All other channels considered as slow (magnet currents, etc) and collected with the maximum rate of 1 Hz. The data is collected by fast (FC) and slow collectors (SC) correspondingly via Ethernet. The collectors put data to the Buffer Manager (BM) [3] for online access. Distributors (DS) read data from the BM according to the stream descriptions provided by the Run Control during the DAQ configuration procedure. The data streams are pushed to the Event Builder (EVB) and further to the Writers (WR). The latter writes data to files on a local disk. The files from the local disk are copied to a huge RAID array and accessible via NFS for the public. The experiments data is usually copied to tape for the permanent storage.

*Deutsches Elektronen-Synchrotron, DESY, Hamburg, Germany

DATA MANAGEMENT

The DAQ data management components control the data flow and guarantee all required data is written to data files and to the tape if required. The components keep track of the written data in order to assure fast data access. The rest of the paper will be devoted to the description of those components.

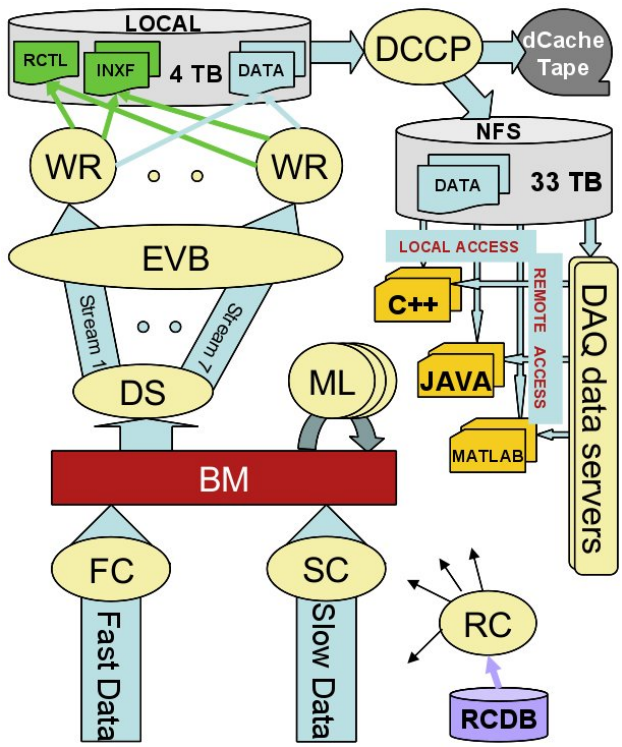


Figure 1: FLASH DAQ dataflow and access tools

Fast Channel Data

The fast channels (~ 700 channels) provide the most part of the data volume (~99.5%). It means that the reduction of the total data amount strongly depends on the configuration of the front-end DAQ senders. The front-end configuration is performed by the Run Control process during the DAQ configuration. The RC is capable to set every parameter for the spectra that are usually sent by the front-end (e.g. start, increment, length). For that the RC has a group of run parameters when changing one of them changes a spectrum parameter in a group of channels (e.g. the same device types). The different sets of run parameters are stored in the Run Modes of the Run Control data base [4]. Every Run Mode corresponds to a

FLASH operations mode. In this way one can control the total amount of written data.

Data produced by Middle Servers (MS) belongs to the fast data. Their configuration is performed by the RC too, and therefore defined by the Run Modes.

Slow Channel Data

The amount of slow channel data is controlled by assigning the channels to two types of slow events: update event and/or environment event. The data for update events is put to the BM periodically (currently every 15 seconds). The channel data for environment events is put to the BM only on its value change. Writing the environment channels can be also controlled by filters (absolute or relative limits for the value difference).

Data Streams

The FLASH DAQ is currently writing 7 data streams simultaneously. Every data stream consists of a set of collected channels. The list of channels for a stream is defined by the experiment using the stream. The largest LINAC stream contains all data collected by the DAQ. The LINAC data can be used by other experiments in case some additional channels are required for their analysis. The stream separation is done by the DS. It receives stream descriptions from the RC during the configuration.

Writers, Run Catalogue and Index Files

Currently EVB is acting as a gateway between distributors and writers. In future one could use it as filter for data streams to reduce the data volume, to generate statistics, etc.

Writer processes are responsible for dumping the data streams into files. The writers keep track of created files by means of a Run Catalogue (RCTL). For every run and stream a set of index files (INXF) is created by the writers. The RCTL is a binary file that contains the start and the stop time for every run and the number of written files. The index files contain the information about every written file. It includes time stamp and event IDs of the first and the last event in the file and the number of events for every event type. The RCTL and INXF allow to find the list of file names for a certain time period and experiment.

Permanent Data Storage

The files written by WRs are shipped from the local disk to the tape storage by the dCache [5] copy process (DCCP). DCCP keeps track of all taped files in a DCCP catalogue.

FLASH DAQ Data Files

We are using a custom designed file format for the data storage. The format is highly optimized for fast data access. The fast access is achieved by writing the data for one channel in a continuous data block (basket). Three steps are usually required to read a channel data:

- Read reference tables and find out the data basket offset
- Read the data basket
- Decompress data if required

Depending on the data type the channel data can be written into the file with or without compression. Two algorithms for compression are supported: ZLIB [6] and LZO [7]. The second one is used in case of CPU power limitations.

DAQ data files are self describing. One can get information about the list of stored channels with their descriptions as well as the number of entries for every channel without accessing the data itself.

DATA ACCESS TOOLS

To be able to work with the DAQ files one needs tools to extract the required data. A set of tools has been developed, providing two different access methods: directly from files (local access) and by means of DAQ data servers (remote access, see Fig.1). In the second case the user software receives the required data from the DAQ data servers running on dedicated computers. The requirements to the data access tools are strongly dependent on the user's task. We have concentrated on the general purpose visualization tools and the libraries that could be used by the experts to make their own processing programs corresponding to their wishes. We have developed libraries and tools for thee environments used at FLASH: C++, Java, MATLAB [8]. Platform supported are Solaris (SPARC), Linux (Debian and Ubuntu) as well as Mac OS X.

C++ BASED TOOLS

A set of classes has been developed to access DAQ data from files. In order to start the data extraction one needs to provide a "data request" containing the time period (start, stop, or a run number), list of channels to extract and the experiment name. One can set all those parameters either by means of the corresponding methods of classes or by providing the name of a XML file containing all required information. Once the request is defined, a method to start the data extraction is to be invoked. Due to multithreading design one can get the required data simultaneously with ongoing data extraction.

Based on the described libraries a data processing framework has been developed. It takes care of the data extraction. The user is provided with 4 routines: user_help(), user_init(), user_loop() and user_end(). The routines can be rewritten by the user, recompiled and re-linked with the framework. The loop routine is called to provide the user code with the channels data in the sequence as it was collected by the DAQ. The init and end routines are called once for user code initialization and finalization correspondingly. The FLASH accelerator and photon experiment groups are using this framework since quite some time to their satisfaction.

JAVA BASED TOOLS

A library (JDAQ) has been developed in Java to provide DAQ data access for Java based applications. The library offers both local and remote DAQ data access. The library contains the same classes as the C++ ones. The same XML configurations can be used for both environments. A few Java GUIs exploding JDAQ have been developed: FLASH DAQ data GUI, FLASH DAQ data Converter, JDDD [9] expert panels.

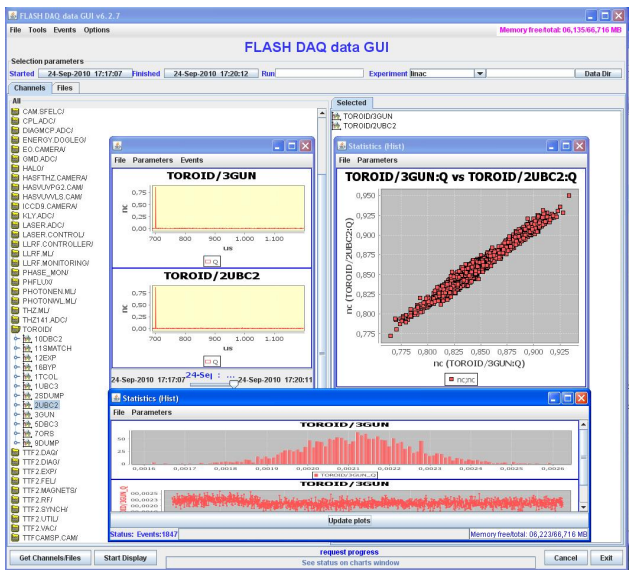


Figure 2: FLASH DAQ data GUI

The FLASH DAQ data GUI (see Fig. 2) is a general visualization tool. It makes use of JFreeChart [10] to draw waveform, histories and histogram plots. The GUI allows watching a set of channels signals as they were during every shot in the linac. One can create histograms and histories for every selected channel as well as the correlation plot for any pair of channels. The GUI can be used for local and remote data access.

The FLASH DAQ data converter dumps the DAQ data to ASCII files. The GUI has a convenient interface to plug-in other converters for producing other data formats.

JDDD is becoming the new display tool for FLASH operators and experts. A few additional components based on JDAQ have been written. They allow drawing stored DAQ data along with online data read from the FLASH control system. Because of that new capability of JDDD one can easily build panels for the experts to analyze the behaviour of their setups in the past.

The LLRF expert coupler interlock panel is a good example of that approach. The expert selects an interlock event shown in the DOOCS [11] history of the beam interlock system. On selection an event a request to DAQ data servers to extract data for a set of channels is sent for 5 seconds period before and after the event. On receiving the requested data the wave forms are plotted in the same plots where the current online waveforms are drawn. The expert can shot by shot go through 10 second period of DAQ data and compare the channels with the online ones. In this way it makes it easy to find out the source of the interlock for the selected interlock event.

MATLAB TOOLS

Based on the C++ classes external MEX functions have been developed to provide access to the DAQ data from within MATLAB. The DAQ data request can be set either by setting an array of strings inside of the MATLAB script or via an XML file with the same format as for C++ and JDAQ libraries. The MEX functions extract the requested data and convert it into MATLAB structures that can be read by MATLAB scripts and analysis code.

PERFORMANCE

The FLASH DAQ currently collects all required beam related channels with the rate of 8000 bunch/s (800 bunches at 10 Hz repetition rate).

The measurement of the data extraction time shows that for modern workstations exploiting fast multi-core processors it mostly depends on disks performance and network bandwidth. In our environment we measure 0.1-0.2 ms/event for reading one spectra channel (2000 floats).

PLANS

We continue to improve our data access tools trying to satisfy our users' requirements. E.g. implementing pre-processing of data by the DAQ data servers could drastically reduce the amount of raw data currently used by the clients and speed up the final data processing.

REFERENCES

[1] http://flash.desy.de/

[2] A. Agababyan *et al.*, "Multi-Processor Based Fast Data Acquisition for a Free Electron Laser and Experiments", IEEE Transactions on Nuclear Science, Vol. 55, No. 1, February 2008.

[3] V. Rybnikov et al., "A Buffer Manager Implementation for the FLASH Data Acquisition System", PCaPAC 2008, Ljubljana, Slovenia, October 2008

[4] G.Dimitrov, "Application of Oracle Database for TTF DAQ System", PCaPAC 2005, Hayama, Japan, March 2005

[5] http://www.dcache.org/

[6] http://www.zlib.net/

[7] http://www.oberhumer.com/opensource/lzo/

[8] http://www.mathworks.com/products/matlab/

[9] http://jddd.desy.de

[10] http://www.jfree.org/jfreechart/

[11] http://doocs.desy.de

BEAM PROFILE MONITORING SYSTEM FOR XFEL/SPRING-8

T. Matsumoto#, A. Yamashita, JASRI/SPring-8, 1-1-1, Kouto, Sayo, Hyogo 679-5198, Japan
S. Inoue, SPring-8 Service Co, Ltd., 2-23-1, Koto, Kamigori, Ako, Hyogo 678-1205, Japan
Y. Otake, RIKEN/SPring-8, 1-1-1, Kouto, Sayo, Hyogo 679-5148, Japan

Abstract

A beam profile monitoring system was developed for XFEL/SPring-8. In this paper, we focus on an image processing system. The image data can be recorded with the synchronized data acquisition system of XFEL/SPring-8. The system is composed of 46 screen monitors (SCMs) and the transverse size and shape of the electron beam are measured down to a resolution of 10 μm. The SCMs provide a valuable tool for beam commissioning in terms of optimization of beam transport and measurement of beam emittance. The imaging system uses CCD cameras that are connected by Camera Link. An image data is selected using the Camera Link selectors and is then processed by an image server. A diagnostic tool for the beam profile monitoring system requires many functions: real-time image monitoring, image analysis, camera control, screen control, etc. We developed a GUI (Graphical User Interface) using Python as a tool to flexibly implement the functions required for the image data. The system was successfully implemented on the SCSS prototype accelerator and it operated as intended. The system can thus be applied to the beam commissioning of XFEL/SPring-8, which is planned for March 2011.

INTRODUCTION

The Japanese X-ray free electron laser (XFEL/SPring-8) is under construction at the SPring-8 site, and its beam commissioning will begin in March 2011 [1]. XFEL/SPring-8 will generate an X-ray laser with a wavelength that is less than 0.1 nm via the SASE (Self-Amplified Spontaneous Emission) process. To achieve this goal, high-precision beam characteristics (a low emittance electron beam less than 1 π mm mrad, etc.) are required, and various types of beam diagnostic tools must be positioned at each stage of the accelerator [2]. In total, 57 RF cavity beam position monitors (RF-BPMs), 49 screen monitors (SCMs) for beam profile measurement, and 35 current transformer (CTs) for beam charge measurement will be installed.

In this paper, we describe an image processing system equipped with 46 SCMs that is used for transverse beam profile measurement with an accuracy of about 10 μm. The remaining 3 SCMs are used for longitudinal beam profile measurement, which will not be addressed here.

The beam profile monitoring system plays the important role of tuning the beam during the beam commissioning. The system is used for optimization of the beam transport and the measurement of beam parameters (emittance, twiss parameters, etc.). In order for the beam commissioning of XFEL/SPring-8 to proceed smoothly, a prototype of the system has been developed and was implemented in the SCSS prototype accelerator to confirm its performance.

SCREEN MONITOR (SCM)

The configuration of an SCM is shown in Figure 1. The SCM system is composed of a screen, a screen actuator, an optical system, and a data acquisition system with a CCD camera. Since the beam is destructed by the screen, the screen actuator moves the screen outside of the beam orbit when it is not used. The material of the screen components was selected depending on the beam energy. For higher energy (>30 MeV), metal foil was used for the optical transition radiation (OTR) while for lower energy (<300 MeV), Ce:YAG was used for fluorescence. In order to achieve a high position resolution, the optical system is equipped with a custom-made lens. The zoom range can be adjusted through the operation of a motor. The position resolution is about 3 μm at a magnification of four times and satisfies a required resolution (10 μm). For equipment controls such as stepper motor controller of the zoom adjustment, Programmable Logic Controllers (PLCs) are used [3]. Two types of CCD cameras are used: a JAI CV-A10 CL (monochrome, 0.46 M pixel, 60 fps) and a JAI CV- M4+CL (monochrome, 1.45 M pixel, 24 fps). For each SCM, the proper CCD camera was selected according to its needs.

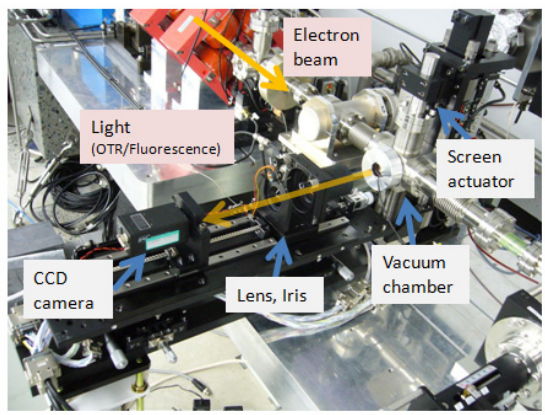


Figure 1: Screen monitor for XFEL/SPring-8.

IMAGE PROCESSING SYSTEM FOR SCM

An overview of the image processing system for SCM is shown in Figure 2. Communication with CCD cameras

#matumot@spring8.or.jp

is performed by Camera Link, which makes it possible to transmit image data, provide trigger signals, and control the cameras. Measurement can only be performed with a camera because the SCM destructs the beam. Therefore, Camera Link selectors (Stack: CLS-900A) are used to select a camera, and the selection signals are inputted into an image server. In total, 11 Camera Link selectors are used to select a signal from among 46 CCD cameras. Cameras are positioned along the accelerator tunnel and the signal cables need to extend into a control room. For this reason, Camera Link signals are converted into optical signals for long-distance transmission.

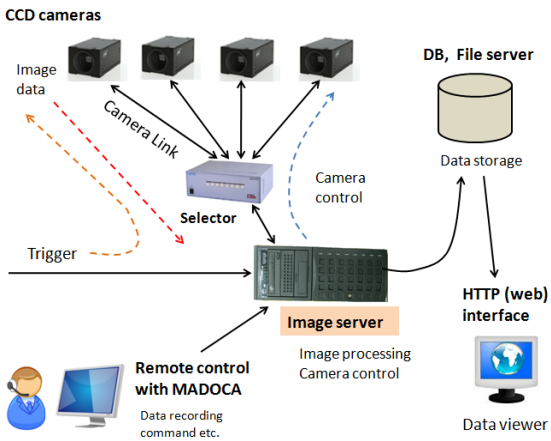


Figure 2: Overview of image processing system.

The image server has two trigger inputs for synchronization of the data acquisition system. One is for a trigger signal that activates the timing of the RF control equipment (60 Hz at maximum, called the RF trigger), and the other one is for a trigger signal for the beam arrival timing (called the beam trigger) [4]. The beam trigger is a sub-divided signal of the RF trigger. The beam trigger is used for data acquisition by the CCD cameras. The RF trigger is used to count identification numbers (called tag numbers) for the synchronization of the data acquisition.

In the image server, an image processing board (AVALDATA APX-3312/1) and a counter board (Interface PEX-632102) are implemented. These are PCI express boards, and Linux drivers are available. Cent OS 5.4 is used as the operating system of the image server. Most of the operations (camera control, image processing, etc.) are performed by the image server, though remote controls can also be applied with a MADOCA framework [5]. Sometimes the image data needs to be taken under changed conditions (the magnetic field values for beam transport, etc.). In such cases, it is useful to be able to operate the data recording remotely.

The data is stored in data storage and can be utilized for analyses. Web interface is available via the data viewer. Due to the large size of the image data, the file names of the image data are saved in the DB (database) and the image data themselves are saved in files.

Software configuration

As shown in Figure 3, the software used for image processing has a multi-layer configuration to intermediate shared memory. Higher level applications (image monitoring GUI, image recorder) send the control commands and receive data (image data, tag numbers, etc.) via shared memory, and a program for image processing and counter boards is operated via the shared memory. Such a system with shared memory has several distinct merits:

- Past image data can be obtained by preparing a buffer area for the image data in the shared memory.
- Multiple processes can utilize image data in the shared memory asynchronously. When the GUI monitors an image, the data can be recorded at the same time.
- Image data other than Camera Link can be easily processed through modification of the program for image processing and counter boards. For example, it was possible to test the system using pseudo data. In future, it will also be possible to process image data for a GbE camera.

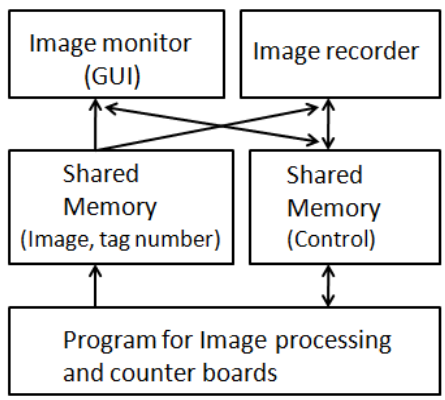


Figure 3: Configuration of software for image processing.

Using Python for GUI construction

Because many functions are implemented in our GUI, its efficient construction is a priority. For this purpose, we implemented Python and WxPython for use in the GUI toolkit. With Python, current existing modules can be easily applied. PIL (Python Imaging Library) and numpy (scientific computing with Python) are especially useful. Several modules (MADOCA control, image analysis, etc.) were prepared for possible use and were utilized to construct the GUI efficiently. In the construction of the GUI, the GUI frame was built using WxFormbuilder and

the layout of the GUI frame was saved in an XRC file (XML format). With the XRC file, the development of the GUI frame and GUI algorithm were kept separate, which simplified the construction process.

GUI FOR BEAM PROFILE MONTIOR

The GUI for the beam profile monitor is used for camera tuning and data recording, and for the image viewer. The image data is recorded with synchronization of the data acquisition system in order to compare the image data with other beam diagnostic data (BPM, CT) from a beam shot. For this purpose, a prototype GUI was developed and implemented in the SCSS prototype accelerator, as shown in Figure 4. The test results, which are described below, confirmed that the system can be applied to XFEL/SPring-8.

Camera tuning

The parameters for the CCD cameras were tuned for exposure time and gain setting before measurements were taken. Tuning can be performed for each camera by selecting a camera using the Camera Link selectors. Screens can be operated in order to see the beam profile image. The tuned values for each camera are stored in the DB and can be loaded at a later time as needed. Background image data can be taken with internal trigger events and utilized for the subtraction. The threshold value for the image data can be also adjusted. After tuning, it is possible to put a lock on the tune values to prevent them from being modified.

Image viewer

Real-time beam profile images can be monitored on the GUI. Beam statistics (center, width, and intensity) and projection histograms can be obtained at the same time, as shown in Figure 4. The monitor rate of 5 Hz is sufficient to enable the beam condition to be evaluated by eye. Beam statistics can be extracted using several slice method options. In the experiment, we initially had trouble seeing the beam shape due to its small size, and therefore developed the GUI to provide expanded image data if necessary. In addition to real-time images, image data stored in the shared memory and image files can also be selected for the monitor in the same manner as above.

Data recording

Image data can be recorded on the GUI. The recording number can be set in advance. To implement the image data in the synchronized data acquisition system, the image data are stored with tag numbers. Since the image server only counts the RF trigger, we need to estimate the offset value in order to extract the tag numbers. To do this, we saved the count number of the RF trigger with a timestamp and compared the data in terms of the relationship between a tag number and the timestamp stored in the DB. We confirmed this procedure with SCS and a reasonable correlation was observed between the BPM data and the beam position obtained from the beam profile image.

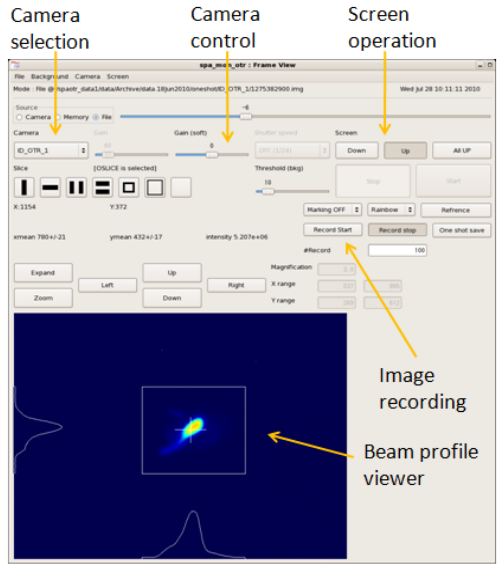


Figure 4: GUI for beam profile monitor implemented in SCSS prototype accelerator.

SUMMARY

A beam profile monitoring system was developed for XFEL/SPring-8. In the system, CCD cameras are connected by Camera Link and the beam profile image is processed by an image server. The image data is recorded with a synchronized data acquisition system in order to be able to see the correlation with other beam diagnostic data obtained from a beam shot. A prototype of the system was tested with the SCSS prototype accelerator, and we confirmed that the system can be applied to XFEL/SPring-8. We are now in the process of constructing the system for XFEL/SPring-8 in preparation for the beam commissioning planned for March 2011.

REFERENCES

[1] T. Shintake et al., “Status report on Japanese XFEL construction status at SPring-8”, Proc. of IPAC’10 (2010).

[2] H. Maesaka et al., “Beam diagnostic system of XFEL/SPring-8”, Proc. of DIPAC’09 (2009).

[3] S. Inoue et al., “Beam monitor system controller for XFEL-SPring-8”, Proc. of ICALEPCS’09 (2009).

[4] M. Yamaga, et al., “Event-Synchronized Acquisition System for SPring-8 XFEL”, Proc. of ICALEPS’09 (2009).

[5] R. Tanaka et al., “The first operation of control system at the SPring-8 storage ring”, Proc. of ICALEPCS’97 (1997).

EMBEDDED CONTROLLER FOR INDUSTRIAL CT TRIGGER MODULE

G. Gong, T.Xue, J.Li, Dept. of Engineering Physics, Tsinghua University, Beijing, China, 100084

Abstract

The industrial CT is used to generate a 3D image of the inside of an object; it consists of an accelerator x-ray source, detector array, readout electronics and control system. A trigger module collects the position information from three decoders installed all the 3 moving axis and generates trigger signal to the x-ray source and readout electronics. The trigger module is remotely accessed by the SCS (system control station) via a fast Ethernet connection. The trigger module utilizes an embedded controller board which consists of a PowerPC controller running the Linux operation system, and a FPGA connected to the PowerPC local bus as a customized peripheral to carry out the trigger logic. With different interface mezzanines and online firmware upgrade, the trigger module has great flexibility to work with different decoders readout electronics.

INTRODUCTION

Originally developed as a medical diagnostic tool, the Computed Tomography (CT) can provide detailed internal information of human body. This technology has also been applied to non-destructive inspect objects that have the indispensable requirement for safety and reliability like high-speed railway train wheels or the air plane turbine engineers. Without the constraints of patient movements or dose restrictions that exist in the medical CT, the industrial CT can achieve better resolution by applying much stronger x-ray source and a much longer exposal time [1].

The industrial CT consists of an x-rays tube, a rotary table, the detector array, the readout electronics, the trigger module and the data analyse and image reconstruction computer.

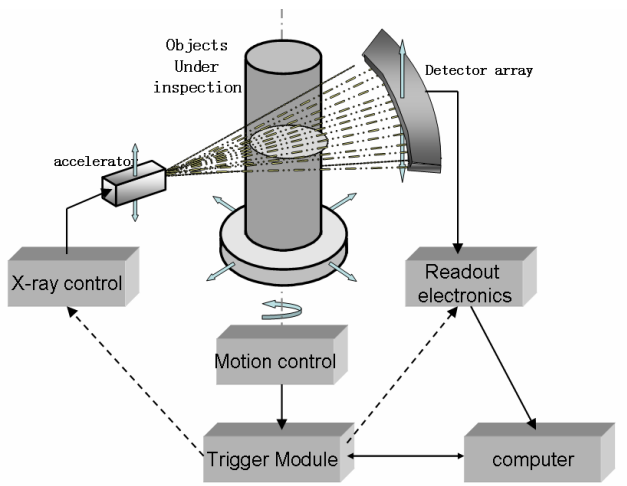


Figure 1: typical schematic block diagram of industrial CT.

A typical schematic block diagram of industrial CT is given in figure 1.

The object to be inspected is located on the rotary table between the x-ray tube and the detector; the X-ray source and the detector are relatively stationary and can move together in the vertical direction along the object; the rotary table can move in two directions and rotate. By setting the relative motion between object and the x-ray/detectors, the industrial CT can be configured to work in direct radiography (DR) mode, second generation CT mode or third-generation CT mode. The x-ray source is working in pulse mode to reduce radiation dose and prolong the life, the readout electronics are also work in gated integration mode to suppress the detector dark noise.

TRIGGER MODULE STRUCTURE

As seen from Figure 1, the trigger module is one of the key components in the industrial CT; it connects with all the other control blocks and manipulates the working flow of the equipment. The functional block structure of the trigger module is shown in figure 2; a picture of the module is given in figure 3.

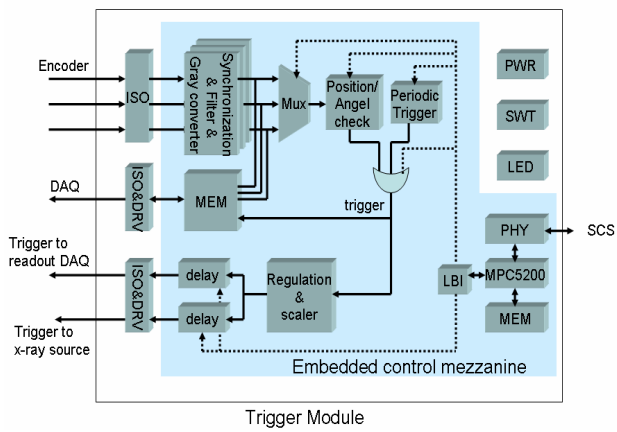


Figure 2: Functional block of trigger module.

Its main function blocks are provided by the embedded control mezzanine which will be described later. A description of all the other elements is given below:

Encoder input

In each axle of the drive motor, there is an absolute rotary encoder installed. They have 16 resolution bits and 8 turns bits to cover the whole scan range. The position and angle the object under inspection can be achieved from the output of these encoders, which are normally in gray code that has only one bit difference between any two consecutive values.

The output of those encoders are converted into the local electrical domain by isolation transistors, a 3 out of 3 low-pass filter removes the noise glitch signals that couple into the cable, then the original Gray code are

converted into binary code for following process.

Two types of encoders are foreseen to be used: the parallel type has a separate signal for each bit; the serial type multiplexes all bits to be sent via a single signal using the SSI protocol. A multi-channel low speed photo-coupler mezzanine is used for parallel type while a single channel high speed magnetic coupler mezzanine is used for serial type. Both mezzanines have the same connector definition thus can be easily replaced.

Trigger output

Depending on the working mode, when the specific encoder outputs indicates that the object has reached a pre-defined position, the trigger module sends a trigger to activate the x-ray source, also sends a trigger to the readout electronics to start integration and conversion after a certain delay caused by the x-ray source latency and detector response time.

The internal trigger logic is working in pipeline mode that can deliver consecutive trigger pulses. Due the limitation of motion system and the minimum trigger width requirement from both x-ray source and readout electronics, the internal trigger pulses are first modulated and pre-scaled before send out. The time delay can be adjusted independently in the step of 10ns for both the x-ray source and readout electronics.

External input

In some product types there are special requirements which need external inputs to the trigger module. For example, in order to inspect the gap of air plane turbine blades, the turbine must be in operation during the inspection. In this case, the motion system is steady but the object itself is rotating, thus the trigger module can be synchronized by the signal from turbine gear control system.

The external input can also used for the safety interlock to avoid unexpected radiation.

DAQ interface

The image reconstruction algorithm needs to know the geometrical position and angel for each acquired tomography slice. Thus the trigger module provides a DAQ interface and transfers the encoders' data to the readout electronics for each trigger. The DAQ electronics packs this information together with detector data to from a complete slice data frame.

The DAQ interface is based on a generally defined high-speed differential serial links. If different DAQ electronics are used in certain product types, an adapter card is needed to convert the serial link to whatever the DAQ electronics asked for.

SCS connection

To configure the parameters and to monitor the status of the trigger module, like to set the work mode or to check the trigger rate etc, the trigger module is connected to the SCS (System Control Station) via the fast Ethernet link provided by the embedded controller.

Switch

Few switches are used to set the encoder types, the DAQ electronics types and other configurations.

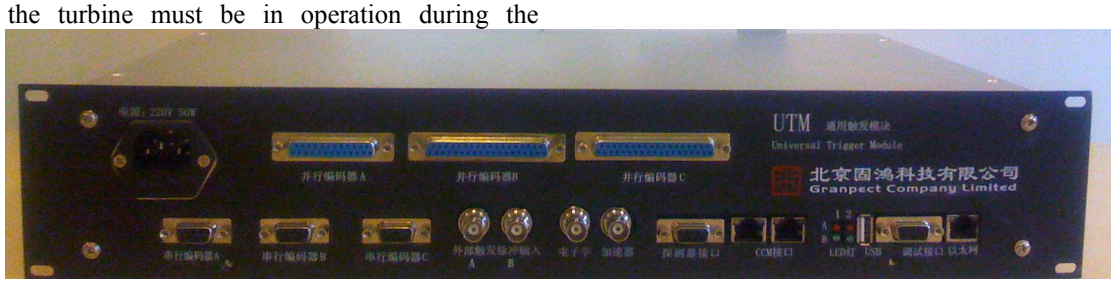


Figure 3: The front panel of the trigger module for industrial CT

EMBEDDED CONTROL MEZZANINE

All the logic and control functions are carried out in the embedded control mezzanine which consists of a processor running the Linux operation system and a FPGA device for the user specific firmware logic [2]. A description of the major items of the mezzanine is given below.

MPC5200 processor

The PowerPC processor from Freescale, MPC5200, with the necessary peripheral, memories and interface devices is the kernel of the embedded controller mezzanine, it has the following features:

- Running at 400MHz
- with 256MB DDR-266 SDRAM
- 16MB NOR flash memory for Linux kernel, file system and user application
- 100Mbps fast Ethernet connection
- Plenty of slow control interfaces like USB, CAN, IIC, SPI.
- Support Linux 2.4 or 2.6 operation system
- boot loader supported

The Linux operation system provides complete support for network protocol stack, file system, device drivers, multi thread communication and many other useful functional packages.

The process FPGA

The fast and real-time trigger process is implemented in a FPGA device, the EP2C35 from Altera. It provides

enough logic elements and distributed memory for the industrial CT trigger logic.

The FPGA is configured in passive serial mode, the raw binary configuration file is stored in the NOR flash. After power up or reboot, the processor executes the u-boot code to read the rbf file and configure the FPGA accordingly. In this way, it is very easy to update the FPGA logic by downloading a new rbf file to the Linux file system via the Ethernet link.

Utilizing the local bus interface logic, The FPGA can be accessed by the PowerPC processor to change the configuration and check the internal information.

SCS client daemon program

A daemon service for the SCS communication is automatically latched after the Linux operation system is booted up. Two Ethernet sockets are created for sending commands and reading information. The daemon program has a simple command line interface that can be easily accessed from any computer via telent, ssh or even simple serial connection.

Self test and diagnostic feature

The embedded control mezzanine has several self test features to check the system integrality and interface connections.

- Each encoder signal is checked for the possible break or short connection. This is very useful for encoder cable and connection diagnostic.
- Self test of the trigger output and external input circuit with a loop back cable.
- A pseudo random counter can be sent through the DAQ interface to check the connection and the transition error rate.

SUMMARY

We have built a universal trigger module for the industrial CT product. The module collects the position and angle information from encoders and performs a trigger algorithm to deliver trigger signal back to the x-ray source and readout electronics. An embedded control mezzanine is utilized in the trigger module, which consists of a FPGA device to perform the trigger logic and a PowerPC processor running Linux operation system to provide Ethernet connection, FPGA configuration, and slow control. The design has shown great adaptability and flexibility for the industrial CT application.

REFERENCES

[1] Hsieh, J., 2003, computed tomography: principles, design, artifacts, and recent advances, SPIE Press monograph, PM114.

[2] XUE Tao，GONG Guanghua，etc, The Design and Realization of General High--Speed RAINl00B DAQ Module Based on PowerPC MPC5200B Processor. NUCLEAR ELECTRONICS & DETECTION TECHNOLOGY, p186, 2010, 30(2)

PRESENTATION ONLY

DATABASE-DRIVEN STATUS ANALYSIS IN BEAM OPERATION AT THE HEIDELBERG ION THERAPY CENTER

K. Höppner*, R. Cee, M. Galonska, T. Haberer, J. M. Mosthaf, A. Peters, S. Scheloske
Heidelberg Ionenstrahl-Therapie Centrum (HIT)
HIT Betriebs GmbH am Universitätsklinikum Heidelberg, Germany

Abstract

The HIT (Heidelberg Ion Therapy) center is the first dedicated European accelerator facility for cancer therapy using both carbon ions and protons, located at the university hospital in Heidelberg. It provides three treatment rooms, two with fixed beam exit (operational since Nov. 2009 and Sept. 2010, respectively), and the first gantry worldwide where the beam exit can be rotated by 360 degrees, currently under commissioning.

HIT uses a PC-based proprietary software system for accelerator controls with an Oracle database for storing device parameters, beam history, error logging etc. Since medical treatment of humans requires a high level of quality assurance, a detailed analysis of beam quality and error logs is needed. We wrote a series of database applications using Python to perform these tasks automatically and create daily reports on beam statistics and parameters, machine status and errors occurred. Additionally, some graphical applications on top of the commercial control system help the scientists and operators in the beam commissioning of the new therapy treatment rooms and the gantry. We will present these applications and show how they are used at HIT.

INTRODUCTION

The HIT accelerator setup as shown in Fig. 1 consists of

- two ion sources, currently used for producing carbon and proton ions (a third ion source is to be installed soon)
- a linac accelerating the ions to 7 MeV/u,
- a synchrotron used to accelerate the ions to their final energy as defined by the patient-specific treatment plan, and
- four high energy beam transport lines providing the beam to the horizontal treatment rooms, the rotatable gantry or the additional station dedicated to quality assurance (QA), research and development.

Cancer treatment with different ion types and of different patients in parallel requires a multiplexed beam operation with the possibility to switch the ion source and beam destination from pulse to pulse, every source/destination combination identified by a virtual accelerator number. Beam parameters can be chosen from a matrix of 255 energy values, up to 6 focus sizes and up to 15 different intensity values, denoted as MEFI parameters. For commissioning and quality assurance, the beam is requested by the Accelerator Control System (ACS) directly, while in therapy mode the Therapy Control System (TCS) requests the beam characteristics determined by the treatment plan via a communication interface to the ACS.

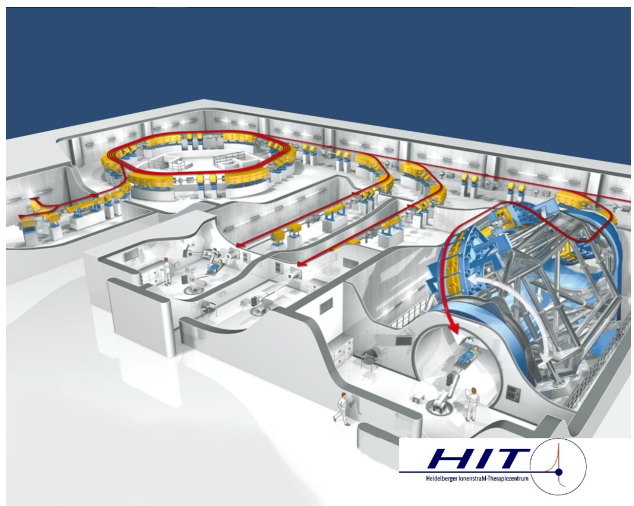


Figure 1: Overview of the HIT accelerator complex. (The QA station is not shown.)

HIT uses an accelerator control system built by a German company for automation and process control hard- and software [1]. It runs on Windows servers, using Oracle 9i as database backend. An upgrade to Oracle 11g is planned for the end of this year. While changes to the GUIs of the control system require invention by the supplier, we can easily access the tablespace used for accelerator controls in the Oracle database. Thus, we were able to develop database applications for various tasks in beam analysis and machine commissioning.

APPLICATION ENVIRONMENT

For a fast development of database applications that may easily be deployed on different systems (Windows for client PCs in the accelerator control room, Linux for a server that is mainly used for the web based electronic logbook [2]), we decided for Python 2.6 [3] as a cross platform OO scripting language. Python provides a variety of builtin and 3rd party modules, including an Oracle module [4] compliant to the Python Database API 2.0. Table 1

* klaus.hoeppner@med.uni-heidelberg.de

Table 1: List of used Python modules

Module	Purpose
cx_Oracle	DB-API 2.0 compliant Python wrapper for Oracle client libraries
csv	reading and writing comma separated values
struct	binary I/O
Datetime	providing date/time funtions
wxPython 2.8	cross platform GUI library

lists the most important Python modules used for our applications.

Since the creation of many data plots is needed both in beam analysis and commissioning, we use Gnuplot 4.4 as graphical utility that can be used in batch mode to produce plots from a list of plot commands automatically created by a script.

While the applications used for commisioning are realized as graphical applications on Windows client PCs using the wxPython wrapper for the C++ wxWidgets library [5], the apps for beam analysis exist either as Windows GUI for interactive analysis tasks or as command line scripts that are run by the cron demon on the Linux elog server to create automatic reports on beam performance.

DATABASE TABLES

The accelerator control system stores any beam cycle with its attributes in a database table:

- Unique cycle id as primary key,
- start and end time of cycle, and
- foreign keys to other tables, like beam mode, cycle status and MEFI combination.

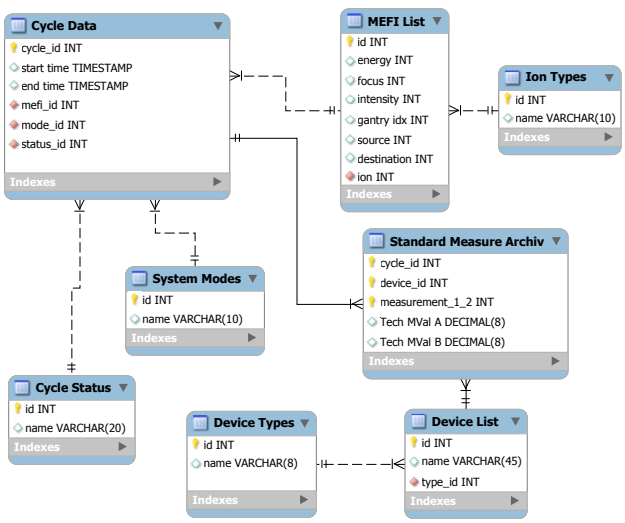


Figure 2: Simplified ER diagram of database tables.

Table 2: Analysis example: used beam request modes

	Cycles: Beam Modes				
	All	Carbon		Protons	
	Cycles	Cycles	Particles	Cycles	Particles
Total	3744	3094	1.36×10^{11}	650	1.53×10^{12}
TCS	2369	1778	3.43×10^{10}	591	1.47×10^{12}
ACS/Exp	1257	1257	1.01×10^{11}	0	0.00×10^{00}
ACS/QS	118	59	1.47×10^{09}	59	5.87×10^{10}

Measured values—like beam position and size on grids or currents of power supplies—are stored in a different table using the combination of cycle id, device id and measurement index (since some devices are measured twice per cycle) as primary key.

A simplified of the database tables used for storing cycle parameters and measured values is shown in Fig. 2.

BEAM ANALYSIS TOOLS

As mentioned before, reports on beam performance are created by a Linux cron job, both on daily and weekly intervall. The cycle data for the time intervall of interest are read from the database and are analyzed by the following criteria:

- Number of failed cycles with analysis of error messages,
- origin of beam request: Therapy Control System (therapy) or Accelerator Control System (technical),
- ion types and destinations of beams,
- distribution of MEFI parameters in therapy mode: energy, focus and intensity.

We use HTML and LaTeX as output formats, the former for a quick overview on beam performance within the electronic logbook, the latter for printed reports. Additionally, all beam parameters are archived as comma separated values since the maximum storage time in database is about six weeks. Thus, a long term analysis (e. g., the number of particles per year has to be checked against the limit in the permission for operation) is possible. Table 2 gives an example for the automatic analysis of the system modes that requested the beam during a 24 h period, while Fig. 3 shows the chronological distribution of virtual accelerators (i. e. combinations of source and destination) during the same period.

TOOLS FOR COMMSSIONING

While patient treatment at a horizontal station started in 2009, the rotatable gantry is still under commissioning [6]. The raster scan technology used at the treatment stations requires a fixed beam position and focus size for all combinations of beam parameters: energy (E), focus (F), intensity (I) and in the case of the gantry also the rotation angle (G).

During commissioning, the beam is optimized for a subset of E, F, I and G combinations that are used as nodes

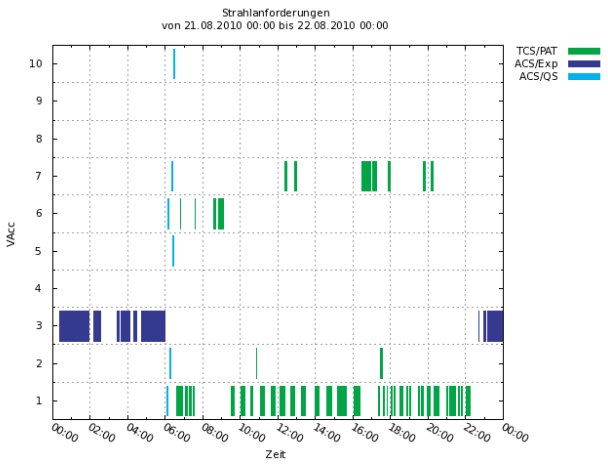


Figure 3: Chronological analysis of cycles. Virtual Accelerators (VAcc) denote a unique pair of ion source and beam destination.

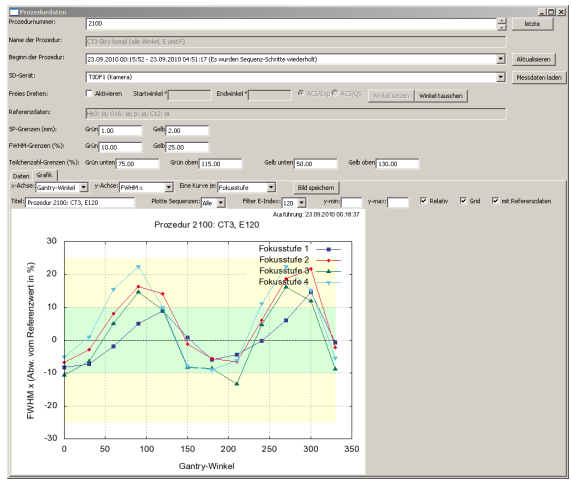


Figure 5: Relative deviation of horizontal beam size from reference value as a function of gantry angle, where a slice of the data set with energy index 120 is plotted.

for an interpolation of the full $E \times F \times I \times G$ MEFI space. The quality of interpolation is checked by running various beam sequences and measuring beam position and size on a fluorescent target. In the past, these checks involved a lot of manual work by the operator, i. e. exporting the data sets as CSV file and importing them into a spreadsheet to get a comparison to the reference values. With our new database applications, we read both the measured values for beam center and FWHM of the beam size and the reference values from the database and directly show the result in a graphical window as table (see Fig. 4) or plot. The GUI supports plotting slices from the $E \times G$ data set, i. e. E on x-axis for a chosen gantry angle G or vice versa. (Fortunately, the set values for the magnets in the gantry don't depend on I.)

The GUI was developed with the aim of supporting the usual tasks performed by the accelerator scientist during commissioning. He can load the data sets from the sequence that was run most recently by a single click and easily chose the device of interest from a list of beam instrumentation devices like the fluorescent target or grids and ionization chambers within the beamlines that were active during the sequence. The standard limits for categorizing the beam quality by colors are read from a configuration file, but these limits may also be changed within the GUI. Plots may be exported as PNG files what is heavily used for adding plot to reports on the outcome of commissioning shifts. For further analysis, data sets can be exported as CSV files.

Figure 5 shows the relative deviation of horizontal beam size from the reference value as a function of gantry angle for a fixed E index.

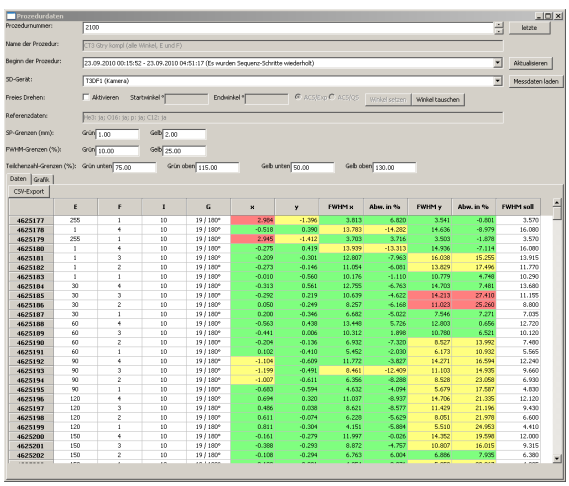


Figure 4: Measured data set: Table view, colors denoting the quality of beam center and focus size, respectively.

REFERENCES

[1] T. Fleck, R. Bär, J. Mosthaf, "Status of the Control System for the Therapy Facility HIT", PCaPAC 2008, Ljubljana, WEP021, p. 215, `http://www.JACoW.org`.

[2] J. Mosthaf, S. Hanke, S. Stumpf, A. Peters, "Using Wordpress as a Simple and Reliable Electronic Logbook for the Heidelberg Ion-Beam Therapy", ICALEPCS 2009, Kobe, THP 111, p. 892, `http://www.JACoW.org`.

[3] Python Programming Language, `http://www.python.org`

[4] cx_Oracle Python extension, `http://cx-oracle.sourceforge.net/`

[5] wxPython, a blending of the wxWidgets C++ class library, `http://www.wxpython.org`

[6] M. Galonska, S. Scheloske, R. Cee, A. Gaffron, K. Höppner, C.M. Kleffner, A. Peters, T. Haberer. "Commissioning of the carbon beam gantry at the Heidelberg Ion Therapy (HIT) accelerator", 7th Workshop on Accelerator Operations (WAO), Daejon, Korea, 2010.

QUARK: A DYNAMIC SDLC METHODOLOGY*

V. Vuppala, J. Vincent, NSCL, East Lansing, MI 48824, USA. #

Abstract

No single Software Development Life-cycle (SDLC) methodology works well for all types of software projects. The project may require a methodology that can be very predictive to very adaptive based on characteristics such as requirements volatility, requirements clarity, project criticality, complexity, and size. We describe a new iterative approach that can vary from being more adaptive to being more predictive during its iterations. The project characteristics change with iterations, and the SDLC adjusts accordingly by changing its parameters. We also discuss the results of using this methodology for projects at National Superconducting Cyclotron Laboratory (NSCL).

INTRODUCTION

Last few decades have seen an evolution of SDLC models to address the software-crisis. Some of these are Waterfall, Spiral, V-Process, RUP, and Agile among others. Each model has its advantages and drawbacks, and not all of them work for all types of software projects [1]. Some of them are predictable in terms of cost and schedule but rigid in terms of requirements, whereas others are adaptive to changes but less predictive.

In our organization there was a need to implement processes to instil engineering rigor into software development. The following were the requirements for the process model:

- Provide transparency and predictability
- Work with limited customer availability
- Not overly bureaucratic, low overhead
- Support project management
- Support critical and non-critical systems

We evaluated various models but found them to be inadequate for our needs. Many organizations, especially in the software industry, choose from a set of SDLC models based on the project characteristics. This was not an option for us, as it required the project team to be proficient in multiple software development methodologies. As a result, we developed a set of processes for software development and project management, which resulted in the Quark Model (QM). It is based on CMMI-Dev 1.2, PMBOK 4, and ISO 9000-3 standards.

Iterations

QM uses an iterative approach to software development. QM iterations are parameterized, and governed by the following parameters (QMPs):

- Duration: The duration, in terms of calendar time, of the iteration
- Change Control: Specification of Major and Minor scope changes
- Documentation: The detail and amount of documentation
- Communication: Meeting intervals and duration within project team, and with Customer
- Planning: Level of detail in planning
- Quality Controls: Frequency of Design and Code reviews, and test methodology.

By adjusting the QMPs, for each iteration, the process can be adjusted from being more adaptive to being more predictive, and anywhere in-between.

Projects

Projects are central to the QM model. A software project is a temporary endeavour undertaken to create a unique software product [2]. It is characterized by certain attributed (PCTs). Some of the PCTs that vary during the execution of a project are:

- Project Team Requirement Clarity: Project team's understanding of the requirements
- Customer Requirement Clarity: Customer's understanding of the requirements
- Size: Size of the project in terms of cost, code base, team size, etc
- Estimate Confidence Level: Accuracy of cost and schedule estimates
- Technology Expertise: Familiarity with the solution technology

Some of the PCTs remain relatively constant during the course of the project, such as criticality of the project, safety and security requirements, quality requirements, timeline constraints, customer Availability, bespoke or custom software, contract type, and team location.

QUARK MODEL

Figure 1 illustrates the Quark Process Model. The PCTs

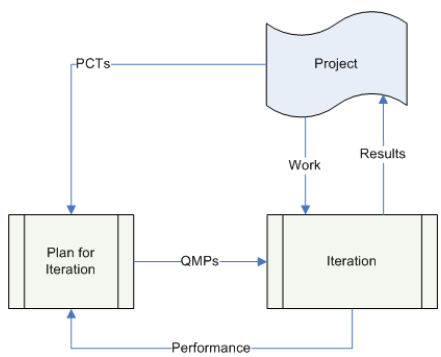


Figure 1: Quark Process Model.

Email Addresses: {vuppala,vincent}@nscl.msu.edu
* This work is funded by National Science Foundation and Michigan State University

and performance (of previous iteration) are used to generate QMPs. The QMPs drive the next iteration, which may result in the modification of the PCTs. Iterations are useful to garner feedback but incur the overhead of test and release management. Hence the number of iterations should be optimized. The idea is to start with shorter duration iterations, and move to longer iterations as the clarity of requirements improves.

Development Process

Figure 2 depicts QM's software development process. It consists of the following major activities:

- Refine Requirements and Architecture
- Plan for iteration or release (PFI)
- Refine design and test plans
- Code, Refactor, Unit Test (CRUT)
- Release
- Deploy and Test
- Review
- Perform User Acceptance Test (UAT)

At the end of each iteration, modifications to the scope, if any, are evaluated. If the change is minor, the next iteration is initiated. However, if the change is major, a Change Request is generated, and the Perform Change Control (PCC) process is initiated. PCC is a Project Management level process, and can result in iteration through the *Plan* process (see below).

In QM, software product goes through release process even for integration tests. This helps with testing of the installation process. Not all releases are sent to the Customer for UAT, and UAT can be proceed in parallel with the execution of next iteration i.e. the next iteration need not wait for feedback from UAT. Configuration management is performed only for production releases.

Project Management

Project Management (PM) is an integral part of QM. Figure 3 shows the QM project management processes with their inputs and outputs. These processes are based on PMBOK-4 [2] but are different especially the *Initiate* process. Goals of the *Initiate* process are to define the scope, develop the solution strategy, and estimate the cost. The results of these activities are documented in Preliminary Project Plan (PPP). PPP is refined in the subsequent process, resulting in the Project Plan (PP). PP also includes the schedule, budget, and plans for quality, risk, communication, and procurement. The level of detail in PP is dictated by the QPMs. The *Execute* process consists of the following activities:

- Acquire and manage the project team
- Conduct procurements, if any
- Perform quality audits (design and code reviews)
- Develop Software using QM Development Process

The *Monitor and Control* process runs in parallel to other activities. It periodically evaluates project performance, procurement status, risks, and quality. It reports project status to stakeholders. The last step in the PM processes is to close the project. Some of the activities here are:

- Obtain Customer feedback and acceptance
- Close procurement activities, if any
- Summarize Lessons Learned, project performance, and customer feedback in Project Closure Report (PCR)
- Archive project related files, and release the team

Project Performance

QM uses Earned Value Management (EVM) [3] to report project performance. EVM is part of the Project Status Report and is measured periodically, generally every week. A Cost Performance Index (CPI) of less than 1.0 indicates that the effort was underestimated, and the project will be over budget if continued at the same pace. A Schedule Performance Index (SPI = EV/PV) below 1.0 indicates that resources were under-allocated, and the project will be delayed. Similarly a CPI of more than 1.0

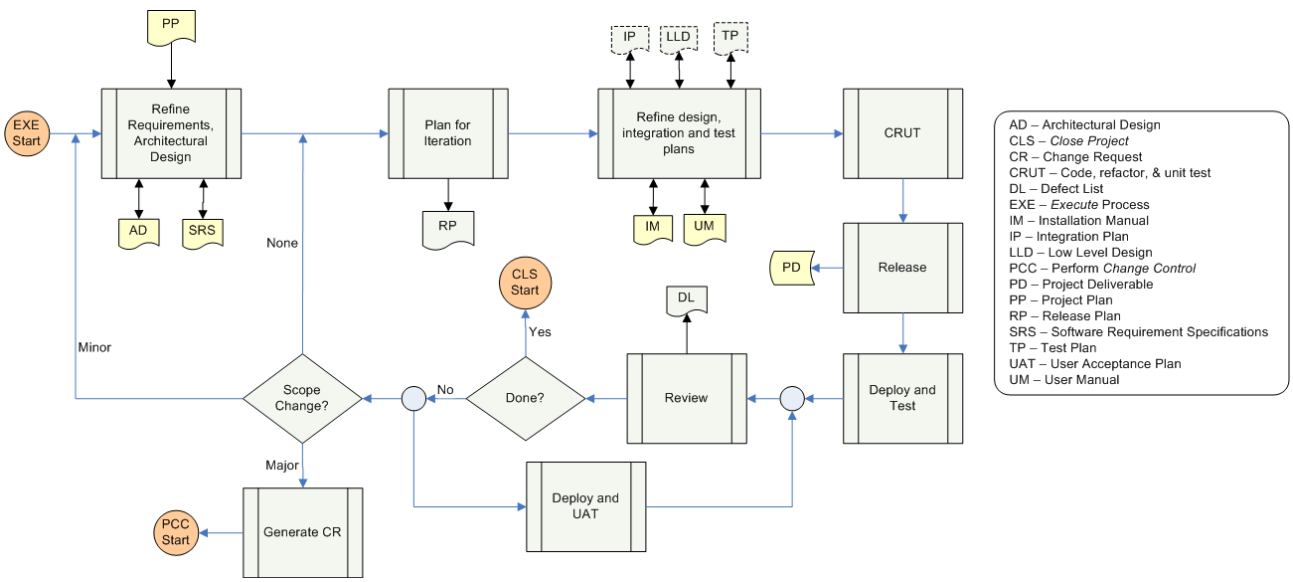


Figure 2: Quark Software Development Processes.

indicates overestimation of effort, and an SPI of more than 1.0 indicates over-allocation of resources. The CPI and SPI values are used to adjust the QPMs for the next iteration.

Documentation

The documents are refined iteratively. QPMs dictate the level of documentation detail. The requirement specifications and the design documents are modified to be in sync with the CRUT activities of the last iteration. This is essential for software maintenance. Some of the required QM documents are Project Plan, Requirement Specifications, Architecture Design, Installation Manual, User Manual, Project Status Report (includes EVM) and Project Closure Report.

IMPLEMENTATION

Based on QM, we have developed the process infrastructure --policies, procedures, guidelines, templates, tools, etc-- for the Electronics Department at NSCL. The process infrastructure is hosted on a website. The project management processes of QM have been generalized, and are being used by non-software groups within the Electronics Department. Currently there are about 5 software development and 15 hardware development projects using the QM processes. All new projects in the department must adhere to the QM processes.

We find that, for software projects, about 8-10% of effort is spent on project management, and a similar amount is spent on documentation. The Customers were very satisfied (9 out of 9) with the ability to make changes, the amount of resources they had to invest, and project management. These results are preliminary; we have not completed enough projects to give a definitive result.

SHORTCOMINGS

QM is not the silver bullet, and has the following drawbacks:

- Currently, measurement of QPMs and the evaluation of PCTs, are subjective. This leaves many decisions to project manager's judgement.
- EVM requires projects to be base-lined, and may not work well for very short iterations.
- It is a slightly heavy-weight model due to the project management processes.

CONCLUSION

Even though QM was developed for our specific needs, it is generic enough to be used by other organizations. Most of the processes, roles, and policies have been designed to be generic; only the guidelines and templates are specific to our environment.

We are currently working on formulating objective measurements of PCTs and QPMs. We are also looking into modifying EVM to suit the Quark Model.

REFERENCES

[1] I. Sommerville, "Software Engineering", 8th Edition, Addison-Wesley, 2007.

[2] ANSI/PMI, "Project Management Book of Knowledge 4th Edition", 2008; http://www.pmi.org.

[3] U.S. Department of Energy, "Earned Value Management". http://www.management.energy.gov/policy_guidance/earned_value_management.htm

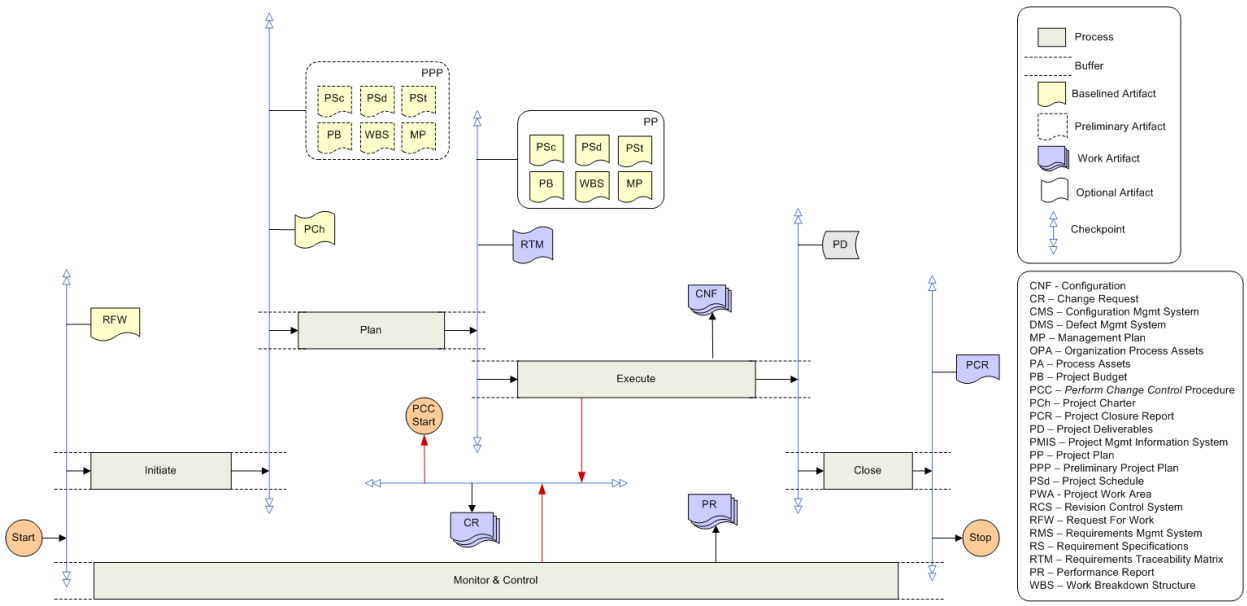


Figure 3: Quark Project Management Processes

EXPERIMENT BASED USER SOFTWARE

D.K. Chevrier, Canadian Light Source Inc., Saskatoon, Canada
M. Boots, Department of Physics and Engineering Physics,
University of Saskatchewan, Saskatoon, Canada

Abstract

The Spherical Grating Monochromator (SGM) and Resonant Elastic-Inelastic Xray Scattering (REIXS) beamlines are located at the Canadian Light Source (CLS). A novel approach to software design has been undertaken to simplify user interactions with these beamlines. While the SGM and REIXS beamlines are structurally different, the techniques available are quite similar. The software is developed to provide seamless acquisition of data, strong data management tools, and easy transition between beamlines for end users. The end result is software focussed on experiments rather than software focussed on beamlines.

INTRODUCTION

One reality of modern science is that 90% of "conducting an experiment" involves sitting at a computer and interacting with software. Traditionally, the CLS has found the resources to develop *beamline software* for each new beamline. In principle, this is a good thing. However, as the facility grows and matures there is a sense that the software used at the beamlines needs to evolve as well. As the vision of the CLS – "[t]o be a global leader and a recognized centre of excellence in synchrotron science and its applications" [1] – makes clear, the purpose of the facility is to support science. As such, evolving our software from *beamline software* to *experiment software* seems like a way to better support science. It is important to note that having *beamline software* is a natural part of the software progression. When a beamline is under development and commissioning, the essential first requirement for software is to provide direct and detailed control over all the separate components that make up the beamline. The importance of this existing software should not be questioned: there would be no way do any science, nor to evolve user software to the next level, had this critical work not been done.

With this background in mind, there are clear ways to address long-standing user issues and improve the experience and efficiency of conducting research at the CLS. The evolution from beamline-centered software to experiment-centered software is accompanied by an evolution from engineering software to designing a *user experience*. That is, there is a shift from the relatively straight-forward task of stating that "software requires the ability to do *functions* A, B, and C using *widgets* X, Y, and Z" to a more holistic need for software that "makes it *quick* and *intuitive* for users to do *tasks* A and B". Because of this change from concrete to descriptive requirements, there are competing types of requirements to keep in mind. In principle the requirements of functionality, appearance, and connectivity will compete with each other as each component is designed and developed. Thus, every component within the software needs to work properly, look appeasing to the user, and be able to connect with other related tasks the user wishes to do.

In addition to a discussion about the concepts and ideas of making an experiment centered software package for users, some time must be devoted to exploring how this can be best achieved from a programming standpoint. While important, the examination of the programming principles will take a backseat to the fundamental vision.

From the inception of this project, we sought to cast as wide a net as possible to determine what users needed out of experiment based software. A summer student was given the task of shadowing users on a number of different beamlines looking for features that were exceptional, tasks that could be simplified, and common irritations that users experienced. Additionally, a workshop was conducted at the CLS Annual Users' Meeting to act as a focus group for new software concepts. A number of outstanding ideas were generated and have been incorporated into the current design.

USER CONCEPTS

Would it not be wonderful if users could sit down and just start doing experiments when they first get to the beamline? Could it be made so software would help users with their experiments – giving them guidance when needed and remaining unobtrusive when not? Would it be so bad if users only needed one software tool from the beginning of their experiment until the end? The vision of experiment based user software is to offer all of these opportunities to users, regardless of their experience level or background, in a way to allows them to concentrate on the science they know. At the same time, the user experience needs to be as pleasant and efficient as possible. The question we must pose is whether it is possible to achieve this and, if it is, how best can that be done? Presuming it is possible, the software evolves from controlling individual acquisitions and beamline actions to managing the acquisition, the data, the beamline, and the experimental process as a whole.

Acquisition Management

Currently, many users experience a steep learning curve when they arrive at the CLS, the steepest part of which is becoming familiar with the unique controls of the beamline they are working on. A common experience might be that of an expert user doing simple x-ray absorption spectroscopy (XAS) at the SGM beamline. Although this user likely knows as much as, if not more than, the beamline staff about the scientific technique itself they are still forced to learn how to conduct XAS on the SGM beamline – which controls to set, how to setup a scan, which detectors to look at, and so forth. Any time a

user changes to a new beamline, the learning curve is repeated. On the other hand, if the technique was explicitly supported in software, two general consequences are expected. First, the lag time stemming from that learning curve could be eliminated as the details of how to coordinate an XAS scan could be programmed for each beamline with no need for the user to get involved if they do not wish to. Second, there would be a seamless transition between the two beamlines; and, not only would the user be able to start immediately, but they would already be familiar with the software. Because of these advantages, technique-based acquisition was one of the first features implemented.

Just as the concept of focusing on the technique rather than the beamline puts the science and the experiment first, the controls for a scan can also be put in scientific terms. Many beamlines will have a chart or a graph posted allowing users to look up the correct set of beamline parameters to achieve, for example, a desired flux and resolution. After choosing the curve they wish to emulate, it is up to the user to move the beamline components to the correct positions. However, since the user was principally concerned with balancing flux and resolution in the first place, could the software not have allowed the user to set these values directly? Furthermore, with appropriate feedback, the user can vary the flux and resolution settings to see what beamline configuration best suits their needs, also giving them a means to learn about the beamline details if they desire. Because of the importance of placing science first, this particular feature has already been implemented.

There are a great number of other concepts that would place science at the forefront as well. Routine users at the CLS are familiar with using a table or similar method to manually define the range of an XAS scan. However, they are primarily interested in scanning their samples for particular elemental edges. While the notion of entirely removing manual entry of a scan range would likely disrupt users, the idea of featuring an interactive periodic table is another way to allow the users to focus science. Since many users need to consult a handbook for the edge energy of the element they are interested in, our goal is to remove the middle-man and allow users to do this direclty in software. Furthermore, such features make it easier for scientists outside of physics and chemistry to use the CLS. This feature, while both important and achievable, is still currently under development.

A key priority for the project as a whole, but with particular focus on acquisition, has been to make the common tasks a user does easy and intuitive. If a beamline has a particular technique that is used more often than the others, or a task that has to be repeated for every technique, then these features have to be designed solidly with great attention to detail and usability.

Finally, while the prior concepts have put emphasis on single acquisitions, it is important to note that all users do many scans while they are at a beamline. Sometimes these scans are done individually with users making decisions between each acquisition; in other circumstances users wish to arrange to do one scan many times, or even to do several different scans in sequence. The concept of a workflow manager is provided to allow users to automate tasks – whether scanning samples, moving between samples, or changing the beamline configuration in some other arbitrary manner. This feature has also been implemented in the initial version.

Data Management

The users of the CLS are accustomed to a process of collecting data; visualizing it in a cursory manner with a limited set of analysis tools; and, finally, transferring all of the raw data to their own computers and re-starting the analysis procedure from scratch. Most beamlines at the CLS offer no tools to assist them in either organizing or logging their experiments before, during, or after acquisition. Common experience shows that almost all groups will record most of the same data by hand into either a logbook or a word processing document. Since this is the case, there is an obvious advantage to having this information automatically collected for the user and stored with their data. Furthermore, since there are no existing tools to help organize data, the addition of a database for scans has been implemented to make it easy for users to sort their data how ever they see fit.

To make the database easy and intuitive to use, many features have been implemented already. Chronological sorting by experiment date and "run" – visit to the facility – is automatic, but users can also create their own *experiments* and sort their scans how ever they wish. This supports users in long-term research conducted across multiple visits to the facility, or across multiple beamlines. A single scan can belong to multiple experiments, if the user wishes, and scans from any run can be placed in any experiment.

A number of familiar user interface paradigms have been adapted to the database so that users can easily identify, select, and organize their large data sets. List views and detailed thumbnail views will soon be implemented to provide additional context and information -- such as beamline configuration -- for each scan. A "logbook" view is also under development: providing a convenient supplement or alternative to paper-based logbooks, and reducing mistakes that users can easily make when recording their experimental process. Finally, drag-and-drop features give users the opportunity to move scans to experiments as well as open scans in the visualization window. Simple features, like selecting multiple scans and collapsible sections, make it even easier to view and move large sets of data.

Along with a database for managing the data users have collected at the beamline, there is also the capability to import data – whether from the CLS or else where. A general structure exists for implementing a new import plugin and, while some coding is required, existing importers for older SGM data and data from Beamline 8 at the Advanced Light Source (ALS) will act as templates. Finally, the importers are optimized to handle large data sets so common users with normal amounts of synchrotron data – the normal amount being "a lot" – can easily see all their historical synchrotron data at once. Of course, if data can be imported in a given format, the software should be capable of exporting as well. One of the short-term goals is to create a framework to do this

efficiently. Not only will it allow users to export the data they have collected at the CLS, it will also allow them to combine data from many facilities and export it all in a common format of their choosing.

Finally, since a user's data should be accessible whether or not they are at the beamline, the data management segment of the software has been designed to be easily separated from the rest of the acquisition tools so users can take it home with them.

Beamline Management

While tools to manage acquisition and user data go a long way to putting science first in the software, there is still the issue that a number of tasks that need to be done regularly on the beamline do not fit into either category. If the aforementioned management systems work as designed, then one of the few remaining barriers to allowing the users to focus almost exclusively on science will be these beamline-specific tasks.

A perfect example of such a task is dealing with samples. Between transferring samples into or out of the chamber, labeling them on the sample plate, and aligning them in the beamline, managing samples can be a substantial undertaking for new, and even experienced, users. Because of these factors, sample management has been given a prominent spot in the initial software design. A central location has been designed to view the sample plate in the beamline; move to and label samples of interest; and, recall sample positions or reload old sample plates. The added benefit of specifying the location of and labeling samples is that the software can automatically associate scans with samples. This association propagates to the database, allowing the user to easily browse by the sample names they have chosen.

In addition to managing samples for acquisition, there is still the matter of transferring samples into and out of the chamber. Like many other beamlines at the CLS, the SGM beamline has a number of manual steps that must be performed to do a sample transfer. Normally, users follow a transfer manual but often have trouble flipping between segments. We are currently testing a software guided manual that allows users to select the transfer task they need to do and gives as much feedback as the beamline has to offer. Furthermore, additions in the near future will add optional pictures, or possibly brief videos, to give additional help as required.

Another beamline task that users often have difficulty with is troubleshooting – particularly determining if the beam's signal strength is appropriate. Normally, users need to ask the beamline scientist which controls to monitor as well as what the feedback value should be. Rather than having users memorize the expected current for different configurations and since beamline characterization has already been done for the flux and resolution settings, another design slated for immediate completion is visual feedback for the signal strength. This simple explanation conjures the image of a cellphone's signal bar, which is exactly how we intend to implement the visual interface.

PROGRAMMER CONCEPTS

Developing this software has presented many challenges: once complexity and interconnectivity reach a certain level programming, undoubtedly, becomes more difficult. However, there has been little doubt that these obstacles could not be overcome – with enough time and code, almost anything seems possible on a modern computer. That being said, we have placed a strong focus on trying to the make the software as easy to code and expand as possible. Some of the design features are discussed in the final sections.

Code Design

As the intention has always been to make the software work across beamlines, the design stresses the use of decoupling and inheritance. The base concepts have been to decouple associated ideas – scan configuration from scan control for instance – and to make "dumb general classes" which are inherited by "smarter specialized classes". Where possible, generalization has taken a backseat to such decoupling and inheritance based on the observation that generalized code tends to do everything in a mediocre fashion while specialized code tends to do one job very well. Our hope is that having the specialized implementation classes completed for a set of beamlines will act as a roadmap for programmers who wish to extend capabilities to their own beamlines.

Code Management

As of September 24th, the project has grown to over 450 files and over 66000 lines of code. Thankfully, the Git version control system has been used to manage the source since the project began. In addition to working well for the initial development period, Git will allow the project to be opened up to a larger community for development – we expect this to happen before the end of 2010. In addition to code management, Doxygen has been used as the documentation suite. Git and Doxygen have been integrated so that the online documentation manual is automatically updated whenever code changes are committed to the version control system.

CONCLUSION

While there remains substantial work to be done, the experiment based user software project has come a long way in a short period of time. By focussing on putting science first and refining the user experience, we hope to deliver software that users enjoy using both at the beamline and when organizing their data at home. With beta testing underway at the SGM beamline, the time is ripe to open the project up to a larger community of contributors, including other CLS staff, CLS users, and collaborators from the larger synchrotron community.

REFERENCES

[1] Canadian Light Source Inc., Retrieved September 2010; http://www.lightsource.ca.

DATA ACQUISITION FROM HETEROGENEOUS SENSOR NETWORKS: THE CASE OF NEPTUNE CANADA THE WORLD'S LARGEST CABLED OCEAN OBSERVATORY

B. Pirenne, Ocean Networks Canada, University of Victoria, BC, Canada

Abstract

Ocean Sciences is at the crossroads: it is entering the brave new world of "Big Science". The first of a new generation of large facilities, the NEPTUNE Canada cabled ocean observatory (www.neptunecanada.ca) will be presented from the point of view of a sensor network composed of hundreds of diverse instruments. The challenges we faced will be reviewed, together with the selected network design, data management and data distribution approaches. Special emphasis will be placed on the architecture of the system and on the more recent developments and concepts used to help scientists in their exploitation of the data. Finally a number of the early discoveries made with the new facility will be briefly described.

CABLED OCEAN OBSERVATORIES

Cabled ocean observatories are remote observing systems that provide power and communication media to a host of underwater instruments and sensors. Consequently, the instruments are (almost) always on-line and sufficient power is provided to the assets to ensure uninterrupted data flow covering multiple environmental parameters at high resolution in a four dimensional space. Observatory systems considered here also provide a significant ability to remotely manage their assets (ie, provide a real-time command ability for specific instruments). As an example, NEPTUNE Canada is composed of a fully redundant 800-km cable loop and has the ability to provide 9kW of power at up to 10 different locations of scientific interest. Figure 1 shows the layout of the NEPTUNE Canada observatory as well as its currently defined 6 main locations, five of which are instrumented. They reside at depth between 20 and 2700 meters.

Each of the locations is equipped with a “node” that reduces the line voltage of 10 kVDC down to 400 VDC and offer data connection points for up to 4 Gbps. In a area covering up to a few km^2, extension cables can be run from the nodes to sites of interest, where platforms with actual instruments and sensors are installed. The platforms are typically composed of a “junction box” whose role is to be the local “power bar”, providing plugs for instrument power and communication, converting the 400 V input to 15, 24 or 48 Volts and translating the instrument serial protocol to IP where necessary.

The instrumentation measures physical and chemical parameters of the ocean (temperature, salinity, oxygen content, CO_2, currents speed and direction at different depths, ...), but also has a number of more specific devices such as underwater video cameras, electro-magnetic experiments, vertical profilers that move through the water column, small vehicles on track (crawler), ... all of which would not be possible without the availability of ample power and the ability to command them in real-time. Figure 2 illustrates the crawler, itself a device equipped with various chemical and physical sensors, cameras, etc.

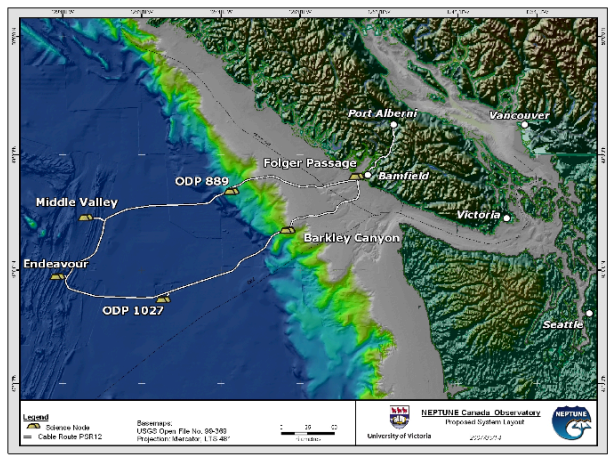


Figure 1: Map of the area covered by NEPTUNE Canada west of Vancouver Island. Please note the 800 km cable loop and the various location of scientific interest, and their "node".

The entire system represent the extension of the Internet under the Ocean, which was the vision put forward by the proponents of such a system many years ago.

Figure 2: A small tethered vehicle on track. It can roam within 50 m from its central position. It is equipped with various physical and chemical sensors and a camera.

NETWORK TOPOLOGY

The network design implements the vision of an Internet-based system, where every instrument and device is either a leaf on the tree structure or a junction point where multiple branches come together. The tree is of variable and arbitrary depth and does not impose conditions on its topology other than the fact that communication to other parts of the network will always propagate up the tree to the first common junction point between any two devices.

Network Design Considerations and Choices

To minimise the cost of the system and to re-use existing off-the-shelf technologies, the use of the Internet Protocol (IP) is preferred as a transport mechanism for data packets at the user/application level. Distances and fibre technology may require another transport mechanism at the lower level. So in this instance the ISO layer 1 can be implemented using fibre optics, lasers and repeaters, on which the SONET protocol will be running. SONET packets will encapsulate layer 2 Ethernet (802.3) packets and deliver them to their end-point thanks to this standard's addressing system. At that level, a traditional network is available for implementing data communication, transport, routing, security, etc.

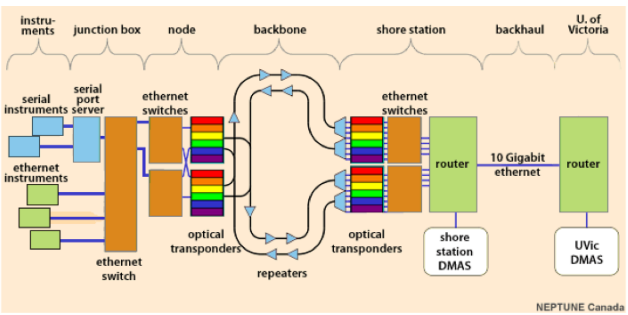


Figure 3: The example of the NEPTUNE Canada network design from a network topology point of view.

As indicated in figure 3 above, currently available oceanographic science instruments are of a legacy design, optimised for power consumption, internal recording and short stays in the water. Their typical data communication interface will be of the serial type (e.g., IEA RS-232, IEA RS-422 or IEA RS-485). To implement the vision of the observatory representing the extension of the Internet underwater, it is necessary to convert the communication protocol of the instrument to IP as close as possible to the instrument. This can be done with simple devices, typically called “terminal servers” enclosed either in the original instrument, in a can on the cable linking it to a junction box or within the junction box itself, often only metres away from the instrument.

To be complete, the structure must also accommodate multiple nodes at the same level, daisy-chained nodes; many junction boxes per node and daisy-chained junction boxes; instruments with piggy-back sensors; possibly multiple shore stations at the root of a network and finally also possibly several redundant data centres.

With a potential for thousands of individual instruments and devices attached to the network, as well as for ease of isolation of the system, it makes sense to select a non-routable set of addresses, as allowed by the IP protocol. In this case, given the complexity of the network, the familiar 10.0.0.0 address space (RFC 1918) was selected. It allows system managers and security analysts to only worry about a few selected bridges between the outside world and the private network, while allowing complete freedom of address allocation and division into VLANs etc. within the private domain.

Virtual Local Area Networks (VLAN – IEEE 802.1Q) offer service segmentation and will be the tool of choice if special categories of instruments need to be isolated from one another for security reasons. VLANs are a layer 2 feature. There are multiple examples that can be considered where VLANs use would make considerable sense in the set up of an observatory. The example of a separate management VLAN comes to mind where all non-user accessible devices will be isolated in a special management VLAN. Such devices will include all network devices on the system (on land as well as underwater) such as switches, routers, media converters, serial-IP converters; but also the facility control computers, precision clocks, etc.

Another VLAN that should be considered is one that will host all instruments that are considered of “national security concern” and would need to be especially protected or have a different management policy.

Timing and time signal

There is a requirement that all clocks on the system be synchronised with a master clock to ensure that all data have the same time baseline to ensure the ability to cross-correlate measurements from different sources. This requirement can be satisfied in a number of ways:

- convince instrument manufacturers to create smart instrument interfaces to periodically re-synchronise the internal instrument clock to the observatory's using the NTP or PTP protocol
- periodically and programmatically re-synchronise the instrument clocks through shore-based software
- time-tag all arriving measurements at the shore station.

Our current approach has been a combination of the first and third option so far, as most of the instrumentation in place is of a legacy, low-power, battery-operated type that is optimised for durability of deployment.

DATA ACQUISITION

A part of the complexity of ocean sciences stems from its plurality: an observatory such as NEPTUNE Canada is serving many different communities with different goals and relying on different types of instruments to achieve their goals: physical oceanographers and chemists will have sensors measuring directly phenomena of interest while biologists will usually rely on proxies to derive populations, species and abundances. This is reflected in the instrumentation that has to be hosted on the system.

Typical instruments will therefore usually fall into one of three categories from a data management point of view:

Table 1: Categories of data streams and instruments

Category	Instrument	Data Format
Scalar	CTDs, chemical sensors, ...	Return lists of values at regular intervals
Complex	ADCP, still cameras, ...	Return n-dimensional matrices on a regular basis
Stream	Video cameras, hydrophones	Return uninterrupted streams of bytes

For the purpose of designing a software system to manage the data flow coming from various devices connected to the infrastructure, a simple approach can be considered where all instruments are considered as sending a stream of data.

At the highest level of abstraction, given the individual duty cycles of each instrument, all categories will, from time to time, return their measurements as a string of bytes. A scalar instrument may be returning the values of its sensors every second for months on end; a still camera may be programmed to take a picture every day, a video camera may be operated periodically and return a rapid succession of images.

At the same time that each instrument can be considered as a producer of a more or less continuous stream of bytes, another way to look at the problem is to see every new stream of bytes as an event that just occurred and for which some specific processing is required.

We assume here a combination of both approaches to deal with the data flow: each instrument produces data in an ad hoc, not necessarily predictable fashion. The (a)synchronous occurrence of a new sequence of data will trigger the execution of a pre-determined set of processing stages, the last of which will be the archival of said stream.

Science Data vs. Engineering Data

Clearly science data collection is the primary goal of any ocean observatory. However, sensors and instruments are attached to an infrastructure that allows them to operate. The infrastructure typically provides power and communication media to instruments and their hosted sensors. So, unless the infrastructure is “somebody else's problem” (such as is the case when all or part of the infrastructure is contracted out to an external organisation, e.g., satellite data transmission), and regulated through a service level agreement (SLA), the organisation operating the facility has to perform and support a potentially significant number of activities having to do with the oversight of the entire system.

The oversight of the system is usually a 24x7 task that involves the monitoring of a large number of subsystems dealing with power and power distribution as well as with data transmission. All of those subsystems will contain sensors that produce engineering data. The engineering data has to be acquired, converted, verified and checked against ceilings and thresholds on a permanent basis. Any value identified as going beyond pre-set bounds will generate alerts to be dealt with by observatory personnel.

In the example of NEPTUNE Canada, nodes and junction boxes, distributing power and communication facilities to science instruments, are equipped with a large amount of electrical and environmental sensors. Such sensors typically return data at the rate of one Hz. It is estimated that the nodes and junction boxes currently connected on the NEPTUNE Canada network will alone produce about 8 TB of raw scalar data per year.

The data are however essential to help predict trends, offer the ability to conduct forensic analysis to understand why an element has failed, etc. An example where trending will help observatory managers extend the lifetime of the infrastructure and establish a priority list for maintenance and recovery is the analysis of the stability of the various ground leak current sensors. Indeed, in seawater, a complete isolation of any power conductor from seawater is essential to prevent corrosion. A slowly increasing leak current (or reduced resistivity to ground) is an indication that something is amiss somewhere and could lead to accelerated corrosion of subsystems. Switching them off early will increase the lifetime of the rest of the system.

Tools have thus to be provided to engineers and “wet plant” system managers to access, examine and react to events happening underwater. The large number of individual sensors that have to be monitored calls for systems that will automatically and constantly verify that all variables remain within their pre-set boundaries. A network management system (NMS) will collect all alerts that come from any subsystem (power or communication) and draw the attention of system operators when they occur. Automating such tasks is essential to limit the operating costs of the infrastructure to a minimum and to avoid the need for a 24x7 coverage of the operation of the system, limiting the service requirement to having personnel on call.

DATA ARCHIVE

Big Science infrastructure is typically designed and built to last between 25 and 50 years: astronomical observatories, large vessels, nuclear reactors, ... after which they have to either be decommissioned or to

undergo significant refurbishment, upgrades and modernisation. The case of an ocean observatory is no exception, but will likely have a life expectancy towards the lower end of the range, mostly due to the lack of experience with such system as well as the harsh and corrosive environmental conditions to which the various elements of the infrastructure are subjected.

Consequently, with funding hopefully in place to support operations during the entire period, the software systems used to acquire and store the data, monitor and control the infrastructure should be sustained and provide access to the sum total of data, information and knowledge accumulated during the complete history of the facility.

This is one of the fundamental requirements of the software system in charge of the observatory and the reason why the underwater infrastructure does not "just" extend the Internet under the Ocean.

Table 2: Life expectancy of different elements of the System

Element	Longevity
High-level design, topology, external environment	Lifetime
Hardware Architecture	10-15 years
Programming language	10+ years
Operating Systems	10 years
Storage Technology	8-10 years
Design of the main software elements	7 years
Operational computers	4-5 years
Storage system	3-5 years

The numbers in Table 2 above indicate the expected life expectancy of the various elements of any large system and illustrate that throughout its lifetime, constant changes and update will have to take place to keep it operating efficiently and economically as, as is often the case, running an ageing infrastructure is more expensive than a timely adoption of new technologies:

- Old hardware will cost more and more to keep running (e.g., keeping lots of small disk drives in operation rather than a few large ones)
- Old software implementation (legacy software) may make it more difficult to find suitable developers who know about the language, OS, etc.
- Novel instrumentation design or radically different ways of using the underwater infrastructure might lead to the impossibility to continue operating with the assumptions that led to the elaboration of the system to that date. (Disruptive technologies).

OPERATION SUPPORT

A large underwater observatory has many physical components. It also represents a facility that has to have a long life time and will therefore host several generations of caretakers. The complexity is so large that it is impossible for a single person or small group of people to remember everything about the system. Examples of essential information abound: installation date and position, date of recalibration of an instrument and the formulae that have to be used for each of its sensors; when the instrument was turned on and off and by whom, ... This information is absolutely critical to understand the data that any instrument produces. Moreover, when dealing with a multiyear archive of data from instruments with a complicated history, understanding that history is necessary for data users to have some trust in the data quality.

The considerations above imply that the amount of information to be recorded, maintained and presented to users about any component of the observatory is tremendous and usually much more considerable than what casual observers would imagine.

DATA ACCESS

Traditionally, data access consists in providing search screens and a result download facility to users. A number of files are downloaded and have to be individually processed by the user, usually in isolation, with local resources and locally developed or installed software. This model no longer works for disciplines where the amount of data is multiplied by a large number of orders of magnitude while the amount of users remains constant. The model that is currently emerging involves a shift away from the search-download-process approach. The concept of Web 2.0 with its participatory approach is calling for something quite different where users use their web browsers to perform all activities related to the scientific process. Some of the differences are as follows:

- On-line collaborations with remote colleagues and students are the norm. Data volumes are so large and so multi-disciplinary that it is often necessary to seek out the support and advice of colleagues in different disciplines to support a particular project execution. The new collaborators may not be co-located and may work at different times but a "work space" is available for all members of a work group to perform all tasks from data search and examination all the way to the redaction of the final paper.
- Searching and sifting through data is done using other criteria and sources of information than previously available such as annotations provided by "crowdsourcing" activities and data from other observatories using interoperability concepts.
- There is little need to download data: data processing facilities on the Grid or in a computer

Cloud are available through privileged links with the archive. Data processing software libraries and templates are available to run against the data. Instead of downloading data, the new concept encourages the upload of new code to run on the server. New code can first be tested, refined and maybe later made available for all to use.

- With compute facilities becoming utilities, with storage capacity available on the network, there is no need to spend money and time maintaining one's own infrastructure. Shared infrastructures are always available at the other end of the high-capacity network.

SOME OF THE FIRST RESULTS

There is no space on such a summary paper to list, explain and illustrate the findings, discoveries and new knowledge acquired through a novel facility such as NEPTUNE Canada. So the author will refer the news posted on the observatory's home page for up-to-date information. The prospects for new findings are very important as such a system has never been built before, as the spacial, time resolution and accuracy of the measurements are increased by several orders of magnitude and that NEPTUNE Canada is supporting no less than five distinct science disciplines (ocean physics, chemistry, biology, plate tectonics and computer science and engineering). Moreover, it is opening the prospect of multi-disciplinary science discoveries.

REFERENCES

[1] C.R. Barnes, M.M.R. Best, and B. Pirenne. NEPTUNE Canada cabled ocean observatory: Final installation and initial results. APEGBC Innovation, March/April issue, pp. 30-33, in press.

[2] M.M.R. Best, C.R. Barnes, B.D. Bornhold, F.R. Johnson, P. Phibbs and B. Pirenne. Live data from the coast to the deep sea: NEPTUNE Canada. AGU Ocean Sciences Conference, Portland, February 2010.

[3] C. R. Barnes, M. M. R. Best, F. R. Johnson, P. Phibbs and B. Pirenne. The NEPTUNE Canada Project: installing the world's first regional cabled ocean observatory. Chapter in book on Ocean observatories, P. Favali et al. (Eds.), Praxis Publishers, in press.

[4] B. Pirenne. Data Management and Archiving System for seafloor observatories: acquisition, archival, analysis, interoperability. Chapter in book on Ocean observatories, P. Favali et al. (Eds.), Praxis Publishers, in press.

PRESENTATION ONLY

PRESENTATION ONLY

List of Authors

Italic papercodes indicate primary authors

Institutes List

ANL
Argonne
- Carwardine, J.
- Kline, D. M.
- Laird, R.
- Lenkszus, F.
- Mooney, T.
- Quock, D. E.R.
- Ross, S. K.
- Shang, H.
- Xu, S.

ANSTO
Menai
- Hauser, N.

ASCo
Clayton, Victoria
- Farnsworth, R. I.
- Grant, A. K.
- Jackson, G. R.
- Owen, A.
- Rhyder, A.

BNL
Upton, Long Island, New York
- Chabot, D.
- Malitsky, N.
- Satogata, T.
- Shen, G. B.

CEA
Gif-sur-Yvette
- Denis, J. F.
- Gougnaud, F.
- Gournay, J.-F.
- Lussignol, Y.
- Mattei, P.

CERN
Geneva
- Arruat, M.
- Cota, E. G.
- Dubrovskiy, A.
- Fernandez, L.
- Locci, F.
- Lucas Rodriguez, F.
- Saavedra, D. G.
- Serrano, J.
- Wlostowski, T.

CLS
Saskatoon, Saskatchewan
- Berg, R.
- Bergstrom, J. C.
- Black, G.
- Boots, M. J.
- Britton, C.
- Carter, L. M.
- Chen, X. B.
- Chevrier, D. K.
- Cubbon, G.
- Duffy, A. M.
- Finlay, C.
- Fodje, M. N.
- Grochulski, P.
- Hormes, J.
- Hu, S.
- Igarashi, R.
- Janzen, K.
- Li, J.
- Liu, D.
- Matias, E.
- Maxwell, D. G.
- Medrano, D.
- Payne, C. G.
- Tanner, R.
- Vogt, J. M.
- Wright, G.
- Zhang, H.
- Zhang, W.

Concordia University
Montreal
- Wang, C.
- Wang, Z.
- Yan, Y.
- Zhao, L.
- Zhu, Y.

Cosylab
Ljubljana
- Bobnar, J.
- Dedic, J.
- Jansa, G.
- Kriznar, I.
- Kusterle, T.
- Plesko, M.
- Rescic, M.
- Sabjan, R.
- Verstovsek, I.
- Zagar, K.

DESY Zeuthen
Zeuthen
- Melkumyan, D.
- Weisse, S.

DESY
Hamburg

- Ayvazyan, V.
- Duval, P.
- Geng, Z.
- Grecki, M. K.
- Grygiel, G.
- Hatje, J.
- Hensler, O.
- Herb, S. W.
- Hoffmann, M.
- Hoffmann, M. G.
- Jezynski, T.
- Kammering, R.
- Kocharyan, V.
- Koprek, W.
- Kube, G.
- Ludwig, F.
- Rehlich, K.
- Rybnikov, V.
- Schlarb, H.
- Schmidt, Ch.
- Schmitz, R.
- Simrock, S.
- Sombrowski, E.
- Weddig, H. C.
- Wilgen, J.
- Wilksen, T.

Diamond
Oxfordshire
- Cobb, T. M.
- Gillingham, I. J.
- Hamadyk, P.
- Heron, M. T.
- Lay, S. C.
- Mercado, R.
- Pearson, M. R.
- Rees, N. P.

ELETTRA
Basovizza
- Asnicar, F.
- Gaio, G.
- Pivetta, L.
- Scalamera, G.

EMBL
Hamburg
- DiCastro, M.
- Fiedler, S.
- Pazos, A.
- Ristau, U. R.

ESRF
Grenoble
- Meyer, J. M.

ESS-S
Lund
- Peggs, S.
- Trahern, C. G.

GANIL
Caen
- Gillette, P.
- Haquin, C. H.
- Lecorche, E.
- Lemaitre, E.
- Philippe, L.
- Touchard, D. T.

GSI
Darmstadt
- Andre, C. A.
- Baer, R.
- Bräuning, H.
- Fitzek, J.
- Fleck, T.
- Haseitl, R.
- Hoffmann, T.
- Kreider, M.
- Lonsing, R.
- Matthies, S.
- Mueller, R.
- Ondreka, D.
- Pfeiffer, D.
- Prados, C.
- Schwinn, A.

HIT
Heidelberg
- Cee, R.
- Galonska, M.
- Haberer, Th.
- Höppner, K.
- Mosthaf, J. M.
- Peters, A.
- Scheloske, S.

I-Tech
Solkan
- Bardorfer, A.
- Beltram, T.
- Kenda, M.
- Kocevar, H.

IHEP Beijing
Beijing
- Wang, C. H.

INFN-Roma II
Roma
- Catani, L.
- Trycz, M. I.

INFN/LNL
Legnaro (PD)
- Andrighetto, A.
- Bassato, G.
- Conforto, N.
- Costa, L.
- Giacchini, M. G.
- Giovannini, L.
- Vasquez, J. A.

IPHC
Strasbourg Cedex 2
- Graehling, P. G.
- Hosselet, J. H.
- Maazouzi, C.

ITER
St Paul lez Durance
- Di Maio, F.
- Wallander, A.

JASRI/SPring-8
Hyogo-ken
- Furukawa, Y.
- Matsumoto, T.
- Yamashita, A.

KEK
Ibaraki
- Furukawa, K.
- Kosuge, T.
- Nigorikawa, K. N.
- Obina, T.
- Odagiri, J.-I.
- Satoh, M.

KURRI
Osaka
- Takamiya, K.
- Tanigaki, M.

Karlsruhe Institute of Technology (KIT)
Karlsruhe
- Cerff, K.
- Mexner, W.
- Spangenberg, T.

MPI/IPP
Garching
- Fischer, R.
- Giannone, L.
- Lackner, K.

NSCL
East Lansing, Michigan
- Vincent, J. J.
- Vuppala, V.

NSRRC
Hsinchu
- Chang, Y.-T.
- Chen, J.
- Chen, Y. K.
- Cheng, Y.-S.
- Chiu, P. C.
- Hsu, K. T.
- Hu, K. H.
- Kuo, C. H.
- Wang, C.-J.
- Wu, C. Y.

National Instruments
Austin
- Cerna, M. D.
- Nagle, J.
- Ravindran, M. M.
- Ruan, Q.
- Schmidt, D.
- Schultz, K.
- Truchard, J. T.
- Veeramani, A.
- Vrancic, A.
- Wenzel, L.

National University of Ireland
University College Cork
- McCarthy, P. J.

ONC
Victoria
- Pirenne, B.

ORNL
Oak Ridge, Tennessee
- Allen, C. K.
- Pelaia, T. A.

RIKEN/SPring-8
Hyogo
- Otake, Y.

SAGA
Tosu
- Iwasaki, Y.
- Kaneyasu, T.
- Koda, S.
- Takabayashi, Y.

SES
Hyogo-pref.
- Inoue, S. I.

SLAC
Menlo Park, California
- Chu, P.
- Corbett, W. J.
- Wu, J.

TUB
Beijing
- Du, Q.
- Gong, H.
- Guan, X.
- Li, J. M.
- Wei, J.

Tsinghua University
Beijing
- Gong, G. H.
- Shao, B. B.
- Xue, T.

UWO
London, Ontario
- Bauer, M.
- Fuller, M.
- McIntryre, S.
- Qin, J.
- Simpson, T. W.

University of Ljubljana, Faculty of Electrical Engineering
Ljubljana
- Kroflic, Z.

University of Saskatchewan
Saskatoon
- Cao, Y.
- Chen, X. B.

Warsaw University of Technology, Institute of Electronic Systems
Warsaw
- Czuba, K.

Participants List

Alan, Duffy
Canadian Light Source Inc.
Canada

Ayvazyan, Valeri
DESY
Germany

Baribeau, Laurier
Canadian Light Source Inc.
Canada

Bassato, Giorgio
INFN - LNL
Italy

Beavregard, David
Canadian Light Source Inc.
Canada

Berg, Russ
Canadian Light Source Inc.
Canada

Bertwistle, Drew
Canadian Light Source Inc.
Canada

Bilbrough, Grant
Canadian light Source
Canada

Black, Gillian
Canadian Light Source
Canada

Bolibruch, Nicholas
Canadian Light Source Inc.
Canada

Boots, Mark
U of S Dept. of Physics
Canada

Britton, Carmen
Canadian Light Source Inc.
Canada

Cao, Yu
Mechanical Engineering
Canada

Catani,
Lucian
Italy

Cerff, Karlheinz
ANKA-KIT
Germany

Chen, Weifeng
Canadian Light Source Inc.
Canada

Chevrier, David
Canadian Light Source Inc.
Canada

Corbett, Jeff
SLAC
USA

Dallin, Les
Canadian Light Source Inc.
Canada

Dotton, Wade
Canadian Light Source
Canada

Du, Qiang
CPHS
China

Dubrovskiy, Alexey
CERN
Switzerland

Duffy, Alan
Canadian Light Source Inc.
Canada

Duval, Philip
DESY
Germany

Finlay, Carl
Canadian Light Source Inc.
Canada

Fitzek, Jutta
GSI
Germany

Fodie, Michel
Canadian Light Source Inc.
Canada

Furukawa, Yukito
SPring-8/JASRI
Japan

Gillingham, Ian
Diamond Light Source Ltd
Great Britain

Gong, Guanghua
Nuclear System Control
and Application Lab
China

Grygiel, Gerhard
DESY
Germany

Hagen, Ulrich
Siemens AG
Healthcare Technology and Concepts
Germany

Haseitl, Rainer
GSI
Germany

Hatje, Jan
DESY Hamburg
Germany

Hensler, Olaf
DESY
Germany

Hoffmann, Tobias
GSI
Germany

Hu, Song
Canadian Light Source Inc.
Canada

Höppner, Klaus
Heidelberg Ion Therapy Center
Germany

Igarashi, Ru
Canadian Light Source Inc.
Canada

Iwasaki, Yositaka
SAGA Light Source
Japan

Jansen, Carl
Canadian Light Source Inc.
Canada

Janzen, Kathryn
Canadian Light Source Inc.
Canada

Johnson, Terry
Canadian Light Source Inc.
Canada

Kammering, Raimund
DESY
Germany

Kline, David
Argonne National Laboratory
USA

Ko, Yung-Chen
National Synchrotron Radiation
Research Center
Taiwan

Kosuge, Takashi
High Energy Accelerator Research
Organization
Japan

Kreider, Mathias
GSI
Germany

Kroflic, Ziga
Faculty of Electrical Engineering,
University of Ljubljana
Slovenia

Larsson, Krister
MAX-lab
Sweden

Li, Mark
Canadian Light Source Inc.
Canada

Liu, Dong
Canadian Light Source Inc.
Canada

Malitsky, Nikolay
Brookhaun National
USA

Matias, Elder
Canadian Light Source Inc.
Canada

Matsumoto, Takahiro
JASRI/SPring-8
Japan

Maxwell, Dylan
Canadian Light Source Inc.
Canada

McKibben, Mike
Canadian Light Source Inc.
Canada

Medrano, Diony
Canadian Light Source Inc.
Canada

Mercado, Ronaldo
Diamond Light Source Ltd
Great Britain

Mexner, Wolfgang
KIT ISS/ANKA
Germany

Meyer, Jens
ESRF
France

Miller, Denise
Canadian Light Source Inc.
Canada

Mooney, Tim
Argonne National Laboratory
USA

Nussbaumer, Rod
TRIUMF
Canada

O'James, Hea
Diamond Light Source Ltd
Great Britain

Payne, Chris
Canadian Light Source Inc.
Canada

Pazos, Andres
European Molecular Biology Laboratory
Germany

Persson, Andreas
MAX-lab
Sweden

Pirenne, Benoit
NEPTUNE - Canada
Canada

Pivetta, Lorenzo
Elettra
Italy

Portmann, Gregory
Lawrence Berkely National Laboratory
USA

Power, Maria
Argonne National Laboratory
USA

Prados, Cesar
GSI
Germany

Qin, Jinhui
University of Western Ontario
Canada

Quock, Deborah
Argonne National Laboratory
USA

Rhyder, Andrew
Australian Synchrotron
Australia

Richards, Jane
TRIUMF
Canada

Ristau, Uwe
European Molecular Biology Laboratory
Germany

Rybnikov, Vladimir
DESY
Germany

Saed, Abu-Ghannam
SESAME
Jordan

Satogata, Todd
Jefferson Lab
USA

Satoh, Masanori
KEK
Japan

Schmitz, Rüdiger
DESY
Germany

Schultz, Kevin
National Instruments
USA

Schwinn, Alexander
GSI
Germany

Shen, Guobao
Brookhaven National Lab
USA

Spangenberg, Thomas
ANKA-ISS-KIT
Germany

Tanigaki, Minoru
RRI, Kyoto University
Japan

Tanner, Robby
Canadian Light Source Inc.
Canada

Touchard, Dominique
GANIL
France

Trycz, Marcin
INFN, sezione Roma TV
Italy

Vogt, Johannes
Canadian Light Source Inc.
Canada

Vuppala, Vasu
NSCL
USA

Wang, Chunhong
2H1P
China

Weisse, Stefan
DESY
Germany

Wilson, Tony
Canadian Light Source Inc.
Canada

Wright, Glen
Canadian Light Source Inc.
Canada

Wurtz, Ward
Canadian Light Source Inc.
Canada

Wysokinski, Tomasz
Canadian Light Source Inc.
Canada

Xu, Shifu
Argonne National Lab
USA

Yan, Yuhong
Concordia University
Canada

Zhang, Hao
Canadian Light Source Inc.
Canada

Zhu, Yongxin
Concordia University
Canada